Ethical Statements

In this paper, we employ LLMs to simulate user-agent interactions. We take care to ensure that the generated data is free from harmful, biased, offensive, or pornographic content. Moreover, all user profiles are entirely fictional, eliminating concerns related to the disclosure of real users' identities, addresses, or contact information.

Details of PAL-Bench

Pipeline Details

Here, we provide more details for each step of the data synthesis process.

Profile   The brief descriptions of users in the domains of work, health, family, and leisure are each composed of 2 to 3 sentences. User personality is described according to the Big Five personality model (McCrae and John 1992), which consists of five dimensions: openness, conscientiousness, extroversion, agreeableness, and neuroticism. For each dimension, we instruct the LLM to assign a value of "high", "medium", or "low". The inclusion of personality information is motivated by the consideration that user preferences can be strongly associated with their personality traits.

Persona   For each month in the user timeline, a minimum of five sentences is required to describe various objective events the user may encounter. The user's requirement types should be summarized as simple, abstract phrases that reflect overall patterns in the user's long-term experiences, rather than incidental occurrences. For each requirement type, both positive and negative preferences should be described based on the user's identity, experiences, and personality. Each preference description should be 4 to 5 sentences.

Situation   We expand the timeline description for each month into 4 to 6 situations. Each situation is described in at least 5 sentences and revolves around 1 to 3 requirement types. When generating the specific situations for each month, all previously expanded situations from the prior month (i.e., the month just before the current one) are additionally provided as input to avoid temporal inconsistencies in situation details.

Experience   To ensure temporal consistency, the prior situation is provided as input when expanding a situation entry into a detailed experience description.

Dialogue Framework   For the "Solutions" part, we adopt a two-stage generation approach. First, without providing any user-specific information, we ask the LLM to generate 8 diverse candidate solutions for each *requirement*. Then, after providing the user's profile and preferences as input, we ask the LLM to select the 2 solutions that best match and the 2 that least match the user's personalized preferences from the candidate solutions. The 4 selected solutions can then be used for dialogue synthesis, while the full set of 8 candidate solutions can be used for the solution proposal task. This approach ensures that negative solutions are still generally relevant to the current requirement. Based on our prior

exploration, if we directly ask the LLM to generate negative solutions that do not match the user's preferences in a single step, these solutions often conflict with the user requirements, which diminishes the significance of personalized preference modeling.

Logs   For log synthesis, we predefine 8 types of logs and their corresponding referenced formats, as shown in Table 6. Each generated log entry should belong to one of these types. Note that our log formats are abstract and flexible, enabling them to accommodate a wide range of user behaviors. For example, as shown in Table 7, the synthesized user performs diverse actions on multiple platforms even under the same "Device Operation" log type. These include navigating on a mobile phone, using the office system on a computer and tracking fitness with a smart band. Therefore, the predefined log formats can preserve the diversity of user behaviors well.

Dialogues   For dialogue synthesis, we predefine 4 dialogue actions each for the user and the assistant, tailored to the service-oriented dialogue scenario (see Table 8). Additionally, we define the transition patterns between these actions to simulate realistic user-assistant interaction dynamics, as shown in Table 9. Specifically, the assistant first clarifies the user's initial query. Once the user confirms their needs, the assistant provides concrete solutions. The dialogue may then continue with the user and assistant discussing these solutions further or with the assistant responding to the user's feedback on the proposed solutions.

Comparison with Other Datasets

Table 10 presents a comparison between our dataset and other existing datasets. The previous datasets are all limited to long-term dialogue scenarios, while our PAL-Set focuses on long-term user–agent interactions that include both dialogue and log records. Besides, each user in our dataset has a longer interaction history (an average of 28.9 sessions and 400.9 dialogue turns) than most other datasets, with the only exception being LongMemEval (Wu et al. 2025). However, it should be emphasized that in LongMemEval, each user actually has very few persona-grounded dialogues, with most additional dialogues sourced from ShareGPT (Zheng et al. 2023) and UltraChat (Ding et al. 2023) that are unrelated to the current user, simply to extend the context length. Although such a setting can be used to evaluate the "needle-in-a-haystack" capability of LLMs, we claim that it is inadequate to evaluate the ability to integrate and abstract long-term behaviors for the same user. Therefore, in our PAL-Set, we do not simply extend the interaction history in this way. Instead, we simulate a person's intrinsically consistent long-term behaviors and interaction dynamics through a carefully designed multi-step data construction process.

Human Evaluation of Data Quality

We conduct a human evaluation of the constructed PAL-Set to assess the quality of the synthesized logs and dialogues. Specifically, we randomly sample 5 users from the dataset and select 10 sessions from each user's interaction records, resulting in a total of 50 sessions. We recruit three students

Log Type	Referenced Content Format
Web Search	"The user searched for <keywords:string>and viewed <content summary>."
Content Publishing	"The user published a <type: post/article/video>on <online platform:string>, with the main content summarized as: <one-sentence summary>."
Content Browsing	"The user browsed a <type: article/product page/video>on <online platform:string>, with the main content summarized as: <one-sentence summary>."
Message Sending	"The user sent a message to contact:string: '<message content>'."
Message Receiving	"The user received a message from <contact/device platform:string>: '<message content>'."
Schedule Management	"The user created a schedule named '<schedule name>' in system:string, time: <start time>- <end time>, location: <specific location>."
Transaction Record	"The user completed a purchase of <type: product/service>via channel:string, product: <name + specification>."
Device Operation	"The user performed a <operation type>operation on <device type>: <specific operation details (may include specific data or numbers)>."

Table 6: Predefined log types and referenced formats.

Timestamp	Log Type	Log Content
2024-08-13 07:30	Device Operation	The user set up navigation on their smartphone, with the destination set to 15th floor, Guanghui Building.
2024-08-13 08:00	Device Operation	The user logged into the market research system on their office computer and viewed the latest data reports.
2024-08-19 19:00	Device Operation	The user browsed a live course on an online education platform, titled "Practical Paths to Transition from Traditional Roles to Emerging Fields".

Table 7: Several logs of the user "0095" in our dataset. Although these logs all belong to the "Device Operation" type, they reflect various types of actions performed by the user on different platforms.

Dialogue Action	Description
User	
Topic Inquiry	The user initiates an inquiry to the assistant about a certain topic, which is usually brief and somewhat vague.
Requirement Confirmation	The user effectively responds to the assistant's requirement inference, confirming their actual requirement.
Solution Discussion	In response to a proposal provided by the assistant, the user does not give a clear positive or negative evaluation of the overall proposal, but instead discusses specific details of the proposal with the assistant.
Solution Feedback	The user expresses a clear positive approval or negative disapproval attitude toward the proposal suggested by the assistant.
Assistant	
Requirement Prediction	The assistant proactively infers the user's implicit requirement behind their inquiry, based on the user's profile or relevant experience.
Solution Proposal	The assistant proposes a solution as a suggestion for the user, based on the user's specific requirement under the current topic.
Solution Discussion	The assistant responds to the user's opinions or inquiries about the current proposal, further discussing the proposal with the user based on the user's profile or relevant experience, and tries to persuade the user to accept the proposal.
Feedback Response	The assistant responds to the user's expressed positive or negative attitude. It is important to note that no other solutions outside those specified in the dialogue framework should be offered here. In particular, when the user expresses a negative attitude, the assistant should only express apology or regret, and should not attempt to recommend other solutions to the user.

Table 8: Predefined dialogue actions of the user and the assistant.

Granularity	Turn Num.	User Action	Assistant Action
Topic-wise	1 turn	Topic Inquiry	Requirement Prediction
	1~2 turns	Requirement Confirmation	Requirement Prediction or Solution Proposal
Solution-wise	0~1 turn	Solution Discussion	Solution Discussion
	1 turn	Solution Feedback	Solution Discussion or Feedback Response

Table 9: Predefined user and assistant dialogue patterns for our dialogue synthesis, where the assistant's dialogue action in the same turn depends on the user's dialogue action.

Dataset	Avg. Sessions/Turns per User	Time Span	Construction Method	Interaction Scenario
PersonaChat (Zhang et al. 2018)	1 / 14.8	-	Crowdsourcing	human-human
MSC (Xu, Szlam, and Weston 2022)	5 / 53.3	a few days	Crowdsourcing	human-human
CC (Jang, Boo, and Kim 2023)	5 / 58.5	a few hours - years	LLM-generated	human-human
LoCoMo (Maharana et al. 2024)	19.3 / 304.9	a few months	LLM-generated	human-human
LongMemEval (Wu et al. 2025)	500 / 5K	a few months	LLM-generated*	user-assistant
PAL-Set (ours)	28.9 / 400.9	a few months	LLM-generated	user-agent

Table 10: Comparison of PAL-Set with existing datasets. * denotes that a large amount of unrelated dialogue data from publicly released user-AI datasets was added in addition to LLM-generated dialogues, resulting in much longer interaction histories.

majoring in computer science and provide them with reasonable compensation. They are asked to evaluate the logs and dialogues in these selected sessions. For the logs, annotators are provided with the user's profile and the corresponding user experiences, and are asked to judge whether the generated logs reflected the key events in the given situation. For the dialogues, annotators are given the user's profile as well as the predefined overall requirements and preferences, and are required to determine whether the user's utterances in the dialogues contradicted the predefined character traits. For each session, annotators are asked to assign a score ranging from 1 to 3, where 1 indicates the generated content did not match the predefined content, 2 indicates a basic match, and 3 indicates a complete match.

The evaluation results show that both logs and dialogues rarely receive a score of 1. The average score given by the three annotators was 2.75 for logs and 2.67 for dialogues. This demonstrates that our proposed data synthesis process can successfully generate personalized and controllable interaction records, resulting in a high-quality personalized long-term interaction dataset.

Dataset License
The PAL-Set will be released under the CC BY-NC 4.0 license[1] and is intended for research purposes only.

Details of Evaluation Setup
Requirement Restatement We use the "Requirements" part of the "Dialogue Framework" generated during data construction as both input and reference output for the task.

The input is the initial vague inquiry (*user query*) corresponding to each topic, while the ground-truth for BLEU score[2] calculation is the user's complete *requirement* description from the framework. The GPT-4 Score focuses on the degree to which the output matches the *implicit needs* specified in the framework. Since the predefined *implicit needs* include two items, we ask GPT-4 (GPT-4o-0513) to assign a score from 0 to 2 for each sample, adding 1 point for each matched item and 0.5 points if only partially matched. To align with the range of BLEU score, we scale the final scores to the [0, 100] interval for reporting.

Solution Proposal We utilize the "Solutions" part of the "Dialogue Framework" in this task. For the "solution generation" subtask, we use the positive solutions from the framework as references to calculate the BLEU score. Since each topic corresponds to two positive solutions, for each sample we take the maximum score computed with the two references. For the "solution selection" subtask, we acquire the scores directly based on predefined polarity labels of the selected solutions. Among the candidate solutions, two are considered positive, two are considered negative, and the remaining four are considered neutral. Each time a positive solution is selected, 1 point is added; each time a negative solution is selected, 1 point is subtracted; otherwise, the score for the selected solution is 0. Therefore, the score range for each sample is [-2, 2]. Finally, we report scores scaled to [-100, 100].

---

[2]We first use the jieba library to segment the original Chinese text, and then calculate the BLEU scores using the sentence_bleu function from the NLTK library.

Multi-turn Dialogue Interaction This task involves three different LLM roles: User-LLM, Assistant-LLM, and Evaluation-LLM. The User-LLM uses the same model as in the data synthesis stage, Qwen-2.5-Max. The Assistant-LLM uses the Qwen-Max-0428 model, while the Evaluation-LLM uses the GPT-4-turbo-2024-04-09 model.

To effectively simulate the predefined user, for each dialogue, we select a topic from the query set and input the user's profile, current situation, and the relevant requirement or preference from the dialogue framework into the User-LLM, instructing it to follow these characteristics in its interactions with the assistant. To ensure the entire interaction process remains controllable and does not get stuck repeatedly discussing trivial points, we predefine the actions for both the user and the assistant in each turn of the dialogue (predefined dialogue actions are shown in Table 5). But we do not restrict the specific content generated. This results in multi-turn user-assistant dialogue instances for each topic.

When evaluating the effectiveness of the interactions, we ask the Evaluation-LLM to compare the dialogue generated by our method with other baselines for the same topic. Considering that the order of the two samples in LLM-based comparative evaluation may influence the results and that a single evaluation may be subject to randomness, we follow the approach of FairEval (Wang et al. 2024). Specifically, each comparison pair is evaluated six times: in three of them, the dialogue generated by our method is presented first, and in the other three, it is presented second. In each evaluation, the model is required to assign a score to each of the two dialogues. We then average the scores for both methods across the six evaluations. The method with the higher average score is considered the winner for that sample; if the average scores are equal, it is considered a tie.

Details of Method Implementation
Implementation Details of H^{2}Memory

We first describe some construction details of our memory framework. For the log graph (M_G) construction, we apply a sliding window approach to each session, as some sessions contain an excessive number of logs. Specifically, we analyze the relations of up to 10 logs with their preceding logs and limit the maximum number of logs provided as input to 20 (i.e., the 10 logs currently being analyzed and the 10 logs immediately before them). Although the direct edges in the log graph built in this manner only capture relatively short-term relationships (within 20 logs), we believe that longer-term relationships can be represented as multi-hop edges in the graph and thus can still be effectively modeled. When further summarizing the background memory M_B based on the log graph, our predefined background aspects include "work", "health", "family", and "leisure".

In the dialogue part of the memory, we need to abstract the multiple topic outlines extracted from M_T into several principles in M_P. When conducting experiments on our proposed PAL-Bench, we cluster all topic outlines from all sessions and set the number of clusters to 8, thereby obtaining 8 requirement types and their corresponding preference principles in M_P. However, the external dataset LongMemEval

adopts a needle-in-a-haystack setting, where only a single evidence session in the long-term history is grounded to the user's persona, and all other dialogues are completely unrelated to the user. Therefore, when experimenting on the "single-session-preference" subset of this dataset, we directly abstract topic outlines extracted from each dialogue to obtain a corresponding principle, without performing topic clustering across sessions.

As for memory retrieval, we directly use the task question as the retrieval query for the question-answer tasks, which include the requirement restatement and solution proposal task in our PAL-Bench, as well as experiments on Long-MemEval. For the multi-turn dialogue interaction task in our PAL-Bench, we add an extra step where the assistant is required to summarize the user requirements expressed in the current context to form the retrieval query. In addition, for tasks or interactions related to requirement understanding, we extract only requirement-related content from the retrieved M_T and M_P memories, without introducing preference-related information. For other tasks or interactions related to solutions or preferences, we include both solution and preference content.

Implementation Details of Memory Baselines

In the "Experiments" section, we report experimental results for a series of long-term memory baseline methods, including basic RAG approaches (*Turn-level* & *Session-level RAG*) as well as previous memory modeling works (*Recur-Sum* (Wang et al. 2025), *ConditionMem* (Yuan et al. 2025), *MemoryBank* (Zhong et al. 2024)). However, the initial designs of these methods were limited to handling long-term dialogue history and did not take into account the logs in our PAL-Bench. To adapt these methods to our scenario for a fairer comparison with our approach, we additionally incorporate some basic log processing steps for them. For the summary-based method "RecurSum", We construct a separate log summary alongside the dialogue summary for long-term modeling, and we also include all logs of the current session directly. For RAG-based methods, we add the top-k logs retrieved from all previous sessions' logs to the input, as well as all logs from the current session. Furthermore, the embedding models, retrieval methods, and retrieval number k used in all baselines are kept consistent with the settings used in our approach.

Details of Experimental Analysis
Additional Experiments on Other Models

To further verify the generalizability of our H^{2}Memory approach across different LLMs and retrievers, we conduct supplementary experiments on two single-turn tasks using Deepseek-V3.1[3] as the LLM and "bge-base-zh"[4] as the retriever. As shown in Table 11, we report the average results over 3 runs for each method. It can be observed that the combination of (Deepseek-V3.1, bge-base-zh) outperforms the (Qwen-Max, paraphrase-multilingual-mpnet-base-v2) pair used in Table 2 on both tasks. This suggests

---

[3]https://api-docs.deepseek.com/news/news250821
[4]https://huggingface.co/BAAI/bge-base-zh

Methods	G-Score	S-Score
Vanilla (w/o log)	25.61	22.48
Vanilla (with log)	39.91	25.46
RecurSum	33.96	34.48
ConditionMem	38.94	29.98
MemoryBank	40.73	36.92
H^{2}Memory (ours)	**49.77**	**40.40**

Table 11: Supplementary results on two single-turn tasks using Deepseek-V3.1 as the LLM and "bge-base-zh" as the retriever. G-Score corresponds to the "Requirement Restatement" task, while S-Score corresponds to the "Solution Proposal" task.

Ours vs Vanilla (with log)	Requirement	Preference
Evaluation-LLM	**45** / 4 / 31	**44** / 1 / 35
Human	**31** / 26 / 23	**41** / 11 / 28

Table 12: Comparison between Evaluation-LLM and human evaluation results on the 80 selected dialogue samples generated by our method and the vanilla method.

that stronger models can further boost performance on PAL-Bench. Besides, our H^{2}Memory approach still outperforms baseline methods significantly ($p < 0.01$), showing strong generalization.

Case Study of Multi-turn Dialogue Interaction

Figure 5 presents a case from the multi-turn dialogue interaction task. Purple blocks indicate the user's personalized information, green blocks display dialogues generated by our method, and white blocks correspond to baseline methods.

In the process of understanding user requirements, although our assistant does not directly hit the user's implicit needs, it is able to make reasonable inferences by considering the user's specific profile and experiences (highlighted in yellow). In contrast, most baseline methods simply repeat the user's question (Vanilla & RecurSum) or ask open-ended questions (ConditionMem). During the discussion of solutions, our method can accurately provide answers that align with the user's preferences (highlighted in blue). However, other baselines mostly offer solutions that the user does not like (highlighted in gray), reflecting a lack of understanding of the user's true preferences. The MemoryBank method, while showing some understanding of the user's background and experiences, still proposes solutions that are not flexible enough, which leads to user concerns.

Human Evaluation for Multi-turn Dialogue Interaction

To verify the effectiveness of using Evaluation-LLM for automated evaluation in the multi-turn dialogue interaction task, we additionally conduct a small-scale human evaluation. We randomly sample 80 topics from the query set of PAL-Set and collect the dialogue generated by our method and the *Vanilla (with log)* method for each topic. We recruit two students and ask them to perform pairwise comparative

evaluations on the 80 dialogue pairs according to instructions similar to those used by Evaluation-LLM. The results from the two evaluators are aggregated by voting, and we compare the LLM evaluation results with the human evaluation in Table 12.

Human evaluation also shows our method generally outperforms the baseline on both dimensions. However, one difference with LLM-based evaluation is that human evaluators tend to judge more dialogue pairs as ties, while our automated evaluation method prefers to distinguish a winner. This is because our automated evaluation uses the average of 6 scores as the judgment basis, so even a small difference in average scores between the two methods will not be considered a tie. In the future, we can explore setting a certain threshold for score differences—e.g., considering comparisons with an average score difference below this threshold as a tie—to make the automated evaluation results closer to human evaluation.

To exclude this factor, we remove the samples judged as ties by both LLM and human evaluators, and calculate the correlation between LLM-based evaluation and human evaluation on all samples where a winner was determined. The kappa values (Fleiss 1971) on the requirement and preference dimensions were 0.54 and 0.62, respectively. According to McHugh (2012), a value in the range $0.40 < \kappa \le 0.60$ indicates moderate agreement, while $0.61 < \kappa \le 0.80$ indicates substantial agreement. Therefore, the LLM-based evaluation method in our task proves to be an effective approach for conducting large-scale assessments.

Profile

Gender: Male   **Age:** 34

Occupation: A sales manager in Eastern China. Travels often for work. Skilled communicator, good at building client trust despite pressure.

Health status: Irregular eating and frequent business meals have led to stomach issues and weight gain. The user is now concerned about his health.

Family members: Married with a 5-year-old son. Tense marriage, often arguing about child-rearing. Parents live far away, rarely in contact.

Hobbies: Work dominates his time. Limited leisure activities. Occasionally runs at the gym to relieve stress, not for enjoyment. No other significant hobbies.

Requirement

Need #1: Needs a time management tool suitable for his work rhythm.
Evidences:
- The user recently started listing core tasks every morning.
- User tries to incorporate time management tips to optimize work planning.

Need #2: Reduce external distractions to focus on important tasks.
Evidences:
- User missed lunch due to an unexpected meeting, disrupting regular eating habits.
- User worked late at a trade show, returning home at 9 PM.

Overall: User wants a time management system for his fast-paced work that handles planned and unexpected tasks, ensuring key deadlines are met.

Preference

Positive: User prefers practical time management tools with clear priorities, is open to digital apps, and wants customized advice for his specific situation rather than general theory.
Solution Example:
- Spends 10 minutes each morning planning core tasks and assigning specific time slots for each.

Negative: Users reject abstract or complex time management strategies, especially those involving psychological analysis. They resist methods requiring long learning periods or significant habit changes. Generic approaches ignoring work specifics, like frequent travel, are seen as impractical.
Solution Example:
- Try the Pomodoro Technique: 25-minute focus periods followed by 5-minute breaks to boost productivity.

[H²Memory (ours)]

user: How can I make my daily work more efficient?
assistant: Do you want to better manage daily tasks like organizing meeting notes and recording client needs, to improve work efficiency and free up time for family and health?
user: That's about right, but more importantly, how can I reduce the disruption from impromptu meetings and sudden tasks?
assistant: Are you looking for a way to prioritize important and urgent tasks while gracefully declining or postponing less important impromptu meetings and sudden tasks to reduce their interference?
user: Yes, that's it. I need a method that allows me to handle important tasks without being disrupted by less important matters.
assistant: You could try setting a 'do not disturb' time each morning, during which you turn off all non-urgent notifications and focus on your most important tasks.
user: Setting a 'do not disturb' time each morning sounds reasonable, but how do I determine which notifications are urgent?
assistant: Indeed, distinguishing between urgent and non-urgent notifications is important. You can quickly assess urgency based on the sender's identity and message content, with those from superiors or clients potentially being more important. Also, you can set up priority notification features for specific contacts on your phone or computer to ensure timely alerts from these key individuals.
user: Setting up priority notifications for specific contacts could indeed help me better distinguish between urgent and non-urgent matters. This advice is quite practical.
…

[Vanilla (with log)]

user: Is there a way to complete my daily work more efficiently?
assistant: Do you feel that you have too many items on your daily task list, making it difficult to complete all work efficiently?
user: Not exactly, I mainly want to find ways to reduce interference from unexpected matters and ensure important work is completed on time.
assistant: I see, you're looking for a method to prioritize important work and better handle unexpected matters, preventing them from affecting the progress of your core tasks, is that right?
user: Yes, that's correct. I do need a method to prioritize important work and reduce disruptions from unexpected matters.
assistant: You could try using the Pomodoro Technique to improve focus, with 25-minute work units dedicated to a single task, during which you don't handle any unexpected matters. This can help you complete important work more efficiently.
user: A 25-minute unit sounds a bit rigid. I often need to handle urgent client calls or travel arrangements, which might not suit my work rhythm.
assistant: I understand your concern. Indeed, the Pomodoro Technique might not be suitable for work environments that require flexible responses to unexpected matters. However, you could consider dividing your work time into larger blocks, for example, focusing on important tasks in the morning and handling unexpected or secondary tasks in the afternoon. This way, you can ensure core work is completed while also leaving time for urgent situations.
user: This method sounds more in line with my work rhythm. I'll try scheduling important tasks to be handled in the morning.
…

[RecurSum]

user: Is there a way to complete my daily work more efficiently?
assistant: Do you feel that your current work task arrangement is a bit overwhelming, making it difficult to efficiently complete each task?
user: That's about right, mainly I want to find a suitable method for myself to plan tasks and reduce distractions.
…
assistant: You could try the Pomodoro Technique, dividing work time into 25-minute work periods and 5-minute short breaks, with a longer break after every four cycles. This can help maintain efficient work state while allowing timely relaxation and reducing distractions.
user: This Pomodoro Technique sounds a bit rigid. I often need to handle unexpected matters and can't guarantee 25 minutes of complete focus.
assistant: I understand your concern. Indeed, the Pomodoro Technique might not be suitable for work environments that require flexible responses to unexpected matters…
…

[ConditionMem]

user: I've been swamped with work lately. Do you have any suggestions on how I can complete my daily tasks more efficiently?
assistant: Are you looking for techniques to help you prioritize important tasks, or do you want to learn how to better arrange daily affairs to improve overall efficiency?
…
assistant: Understanding that your time is extremely valuable, you could try the Pomodoro Technique. Work focused for 25 minutes, then take a 5-minute break. During these short breaks, you can handle urgent messages or do quick stretches to relax. This ensures efficient work progress while allowing some ability to deal with unexpected situations.
user: The Pomodoro Technique doesn't sound suitable for me. My work rhythm simply can't guarantee fixed focus times.
assistant: I understand your concern. Indeed, the Pomodoro Technique might not be well-suited to your work rhythm…
…

[MemoryBank]

user: Is there any way I can complete my daily work more efficiently?
assistant: It seems you have many important work tasks recently, such as preparing year-end assessment materials and summarizing client performance. Are you hoping to find a way to better manage these tasks and reduce some pressure?
…
assistant: You could try setting fixed meeting times, like dedicating a specific time slot each day for meetings, and focusing on tasks during other times. This can reduce unnecessary meeting disruptions while ensuring enough time to deal with unexpected situations.
user: Fixed meeting times sound good, but what about urgent client issues?
assistant: We can set a high-priority time slot outside of fixed meeting times specifically for handling these unexpected situations…
user: This method sounds feasible, but I'm worried that the high-priority time slot might become too long and affect the completion of other tasks.
…

Figure 5: Case study of the "Multi-turn Dialogue Interaction" task. The text highlighted in yellow and blue demonstrates that our method enables the assistant to more concretely incorporate the user's personalized background in inferring requirements and to provide solutions that align with the user's positive preferences. In contrast, the text highlighted in gray indicates that most of the baselines in this case propose solutions that contradict the user's preferences, reflecting a lack of ability to correctly understand the user's characteristics.

Prompts

In this section, we list all the prompts used in this work, including those for the data synthesis process, task execution and LLM-based evaluation, and the implementation of our method. Note that we present the version of direct English translations from the original Chinese prompts.

Prompts of PAL-Set Construction

Step 1: Profile

You are an excellent data generator, capable of simulating a large amount of user data according to my requirements.

Please generate the basic information for multiple users as required, including the user's name, gender, age, occupation, health status, family members, hobbies, and personality. Specifically:
- Occupation, health status, family members, and hobbies should each be described in 2–3 sentences.
- Personality should be described according to the Big Five personality traits, divided into Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. Each dimension can take one of ["low", "medium", "high"].

Example Data
"""
[
    <example_json>,
    ...
]
"""

Now, please generate 10 basic user profiles, covering as many different user groups as possible (but do not include students or retired individuals). You should refer to the sample data format above and output in JSON format (output user id range: "<start_id>" ∼ "<end_id>"). Please note: do not default to everyone having a harmonious family atmosphere, do not default to everyone being in good health, do not default to everyone having rich and interesting hobbies, and do not default to everyone having generally open, friendly, and positive personalities. In real situations, different people show great differences in all aspects, and you should reflect these differences as much as possible in the simulated data.

Step 2-1: Timeline

Assume the current date is <current_month>1st. Based on a user's basic information, please generate an overview of the user's various experiences over the past several months.

User Basic Information
"""
<user_background>
"""

Output Format
"""
{
    "<start_month>": "...", // Describe the user's experiences during this month
    "<second_month>": "...",
    ...
    "<last_month>": "..."
}
"""

Now, based on the user's basic information, please write about the user's experiences over the past several months (<start_month>∼<last_month>), including various events that may have occurred in their work, health, family, and leisure life. Try not to reflect the user's personality or specific preferences; focus on objectively describing the various events the user has experienced. Note: Please output in the JSON format specified above. Each month's content should be no less than 5 sentences, and there is no need to mention the user's name—just use "the user" to refer to them. Each month's experiences do not have to cover all four aspects (work, health, family, leisure), but should not include only one aspect.

Step 2-2: Requirement Types

Consider scenarios where the user interacts with a personalized daily assistant. The user may seek advice from the assistant on issues related to their own background or experiences; these questions reflect the user's daily requirements. Now, given a user's basic information

and an overview of their experiences, please generate the overall requirement types that this user may have in the areas of work, health, family, and leisure.

User Basic Information
"""
<user_background>
"""

User Experience Overview
"""
<timeline>
"""

Output Format
"""
```
{
    "work": {
        "work-1": "...",
        ...
    },
    "health": {
        "health-1": "...",
        ...
    },
    "family": {
        "family-1": "...",
        ...
    },
    "leisure": {
        "leisure-1": "...",
        ...
    },
}
```
"""

Output Requirements
1. Follow the JSON output format above and generate possible overall requirement types for the user in the areas of work, health, family, and leisure. For each aspect, generate 4–5 different requirement types. The number of requirement types for each aspect does not need to be exactly the same.
2. Each requirement type should be briefly summarized with a short phrase. The descriptions should be relatively abstract and able to cover the user's long-term experiences. Do not be too specific, and do not include occasional requirements tied to a particular moment or specific situation.
3. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Now, following the above requirements and output format, please generate the user's requirement types for each aspect.

Step 2-3: Global Preferences

Consider scenarios where the user interacts with a personalized daily assistant. The user may seek advice from the assistant on questions related to their own background or experiences; these questions reflect the user's daily requirements. In response to the assistant's suggested solutions, the user may express acceptance or rejection, which reflects the user's preferences regarding the solution. Now, given a user's basic information, experience overview, and personality, as well as several requirement types for a certain aspect, please generate the user's preferences for each requirement type.

User Basic Information
"""
<background>
"""

User Experience Overview
"""
<timeline>
"""

User Personality
"""
<personality>
"""

User Requirement Types (in the <aspect>aspect)
"""
<requirement>
"""

Specific Requirements and Output Format
Specific Requirements
1. Following the JSON output format provided below, generate the user's preferences for each given requirement type in the specified aspect. First, copy the requirement type name under "requirement". Then, under "analysis", analyze the user's likely preferences for the requirement type in a paragraph, considering the user's identity, experiences, and personality. Finally, under "preference", clearly provide positive ("pos") and negative ("neg") preference descriptions for the user.
2. In the "analysis", focus on the "User Basic Information" and "User Personality" sections, combining these to analyze the user's possible preferences and the reasons for them with respect to the given requirement type. Use the "User Experience Overview" section only as reference to ensure that the generated user preference descriptions do not obviously conflict with this section.
3. In "preference", provide both positive and negative preference descriptions. The positive preference description represents the types of solution suggestions the user tends to accept when facing this type of requirement, while the negative preference description represents the types of solution suggestions the user tends to reject. Each preference description should consist of 4–5 definitive sentences. The descriptions should highlight the user's personalized characteristics; that is, the suggestions in the positive preference description should not be those that everyone tends to accept, and the suggestions in the negative preference description should not be those that everyone tends to reject as inappropriate.
4. Do not include the user's name in the output; use "the user" to refer to them.
5. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Format (JSON)
"""
```
{
    "<requirement_id>": {
        "requirement": "...", // The corresponding requirement type
        "analysis": "...", // Analysis of the user's likely preferences for this requirement type, considering their identity, experiences, and
personality
        "preference": {
            "pos": "...",
            "neg": "..."
        }
    },
    ... // Generate preferences for each requirement provided
}
```
"""

Now, following the above requirements and output format, please provide the user's preference generation results.

Step 3: Situation

Consider scenarios where the user interacts in a conversational way with a personalized daily assistant. The user may seek advice from the assistant on topics related to their own background or experiences; these topics reflect the user's daily requirements. Now, given a user's background information, the user's overall requirement types{, the user's specific requirement situations in previous months (<last_month_str>)}, and an overview of the user's experiences in the current month, you are required to expand the overview of the user's experiences in the current month into multiple requirement-oriented user situations, and list the corresponding requirement types for each situation.

User Background
"""
<background>
"""

User Requirement Types
"""

"""

User Requirement Situations in Previous Months (<last_month_str>)
"""
<last_month_situation>
"""

Overview of User's Experiences in the Current Month (<cur_month_str>)
"""
<cur_month_summary>
"""

Specific Requirements and Output Format
Specific Requirements
Based on the user's overview of experiences in the current month (<cur_month_str>), please generate 4–6 requirement situations for the user. Each requirement situation should include its corresponding time span ("time_span"), a list of requirement type IDs ("requirement_ids"), and the detailed situation description ("situation"). Details are as follows:
1. The "User Background" in the input can be seen as the user's long-term characteristics{, while "User Requirement Situations in Previous Months" can be seen as the user's short-term historical situations}. These can be used as references to avoid contradictions with the user's history, but the main focus should be on the experiences reflected in the "Overview of User's Experiences in the Current Month".
2. The time span ("time_span") includes the start and end date for the situation, in the specified date format. The time spans for different situations should be consecutive; that is, the start time of the next situation should closely follow the end time of the previous situation. {Note: The start time of the first situation generated here should closely follow the end time of the last user requirement situation from the previous month (<last_month_end_date>).} The end time for all situations generated this time should fall within the current month (<cur_month_str>), and the end time of the last situation can be any day in the last three days of the month.
3. The requirement type list ("requirement_ids") should list the requirement type IDs relevant to the current situation. Each situation may correspond to 1–3 requirement types (if possible, try to include more than one). The requirement types included in the same situation can come from the same aspect or different aspects.
4. The situation description ("situation") should be a paragraph of at least 5 sentences describing the relevant background experiences that led the user to have the listed requirement types. The user's name should not be mentioned in the situation; use "the user" instead. The situation description should cover the span of the time period as much as possible, and should not be overly focused on a single time point.
5. There is no need to emphasize the time range in the content of each generated situation—just describe the situation for that time period directly.
6. Follow the output format below and output in JSON format.
7. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Format
"""
[
    {
        "time_span": "YYYY-MM-DD ∼ YYYY-MM-DD", // Provide the time span in the specified date format
        "requirement_ids": [...], // List the relevant user requirement type IDs
        "situation": "..." // A paragraph of at least 5 sentences
    },
    ... // Generate a total of 4–6 situations
]
"""

Now, please generate and output the user's requirement situations for the current month according to the above requirements and output format.

Step 4: Experience

Given a user's background information, the user's situation in the previous period, and the user's situation in the current period, please expand the user's current situation into a more detailed description of the user's experiences.

User Background
"""
<background>
"""

User's Previous Period Situation (<last_situation_start_timestamp>∼ <last_situation_end_timestamp>)

"""
<last_situation>
"""

User's Current Period Situation (<cur_situation_start_timestamp>∼ <cur_situation_end_timestamp>)
"""
<cur_situation>
"""

Output Requirements
Please expand the user's current situation into a detailed description of the user's experiences for the corresponding time period, with the following specific requirements:
1. The "User Background" in the input can be considered the user's long-term characteristics, while the "User's Previous Period Situation" can be considered the user's recent historical situation. These can be used as references to avoid contradictions with the user's history, but the main focus should still be on the experiences reflected in the "User's Current Period Situation".
2. The generated user experiences should cover the specified time period as much as possible, rather than being completely concentrated on a single point in time.
3. In the generated description of the user's experiences, any subjective intentions, awareness, thoughts, feelings, etc. that may be present in the original situation should be removed. Only the objective events of the user's experiences should be retained and the details of these objective events should be expanded.
4. The generated description of the user's experiences should be specific to the user's daily experiences; there is no need to overemphasize the specific times, but it should better reflect the logical connections between different events.
5. Assume that the user begins to interact with the conversational assistant to seek advice regarding their needs immediately after the end time of the current situation (<cur_situation_end_timestamp>) (there is no need to mention this interaction process in the description). Therefore, the user's experiences prior to the end of the situation should logically avoid contradicting this interaction (e.g., avoid stating that the user started sleeping before the end of the situation), and should also avoid describing events that occur later than the end time of the situation. Note that the times specified in the input are in 24-hour format.
6. Do not mention the user's name in the generated content; use "the user" instead.
7. Do not generate any other analysis or explanatory information apart from the required description of the user's experiences.

Now, according to the above requirements, please directly provide the detailed description of the user's experiences during the current period (<cur_situation_start_timestamp>∼ <cur_situation_end_timestamp>):

Step 5-1: Requirement in Dialogue Framework

Consider a scenario where the user interacts in conversation with a daily assistant. The user may raise questions to the assistant on topics related to their own background or experiences. The assistant needs to use the user's personalized information to understand the implicit intent behind the user's questions, in order to comprehensively understand the user's true current needs. Now, given a user's background, recent situation, and experiences, and specifying the user's current requirement type ("requirement_type"), you are required to, for each requirement type, first simulate the user's initial query ("user_query"), then, combining the user's background and experience, provide the user's implicit intents ("implicit_needs"), and finally summarize the user's specific requirement ("requirement").

User Background
"""
<background>
"""

User's Recent Situation
"""
<situation>
"""

User's Recent Experiences
"""
<experience>
"""

Specific Requirements and Output Template
Specific Requirements
1. Output according to the JSON format template provided below. The template will specify 1–3 topics, and under each topic the user's requirement type ("requirement_type") is already specified. For each topic, you need to first generate the user's initial query ("user_query"), then generate the user's implicit intent list ("implicit_needs"). Each item in the list should include a specific implicit intent description ("need") and the corresponding evidence ("evidences"). Finally, you need to summarize the user's query and implicit intent,

and provide a detailed description of the specific requirement for the current user ("requirement").

2. The user's initial query ("user_query") should reflect the user's query about the specified requirement type under the current situation. This part should be as brief and vague as possible, and should not include too many details from the user's background or experiences.

3. Each user's implicit intent list ("implicit_needs") should contain 2 items; do not add or remove items. The implicit intent description ("need") is something the user did not explicitly mention in their initial query, but which the assistant should understand as the user's intent in the context of the current topic. Each implicit intent description should focus on one aspect and be summarized briefly. The corresponding evidence ("evidences") should be a list, listing 1–5 details from the user's background and experiences that reflect this implicit intent. Each item in the evidence list should only include the relevant detail from the background or experience itself—do not describe the corresponding date, and do not include any additional analysis or reasoning. In addition, each implicit intent in the user's implicit intent list should closely revolve around the requirement type specified by the current topic, and together they should form a focused real need, rather than reflecting different needs across multiple areas.

4. The detailed description of the user's requirement ("requirement") should be based on the user's initial query and implicit intent, and should comprehensively and specifically describe the user's actual need in the current topic. This content should still closely revolve around the specified requirement type, and should not include too much content deviating from that requirement type.

5. Do not include the user's name in the output; use "the user" instead.

6. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template to Complete (JSON format)
"""
<output_template>
"""

Now, following the above requirements and output template, please provide the generated result.

Step 5-2: Candidate Solutions in Dialogue Framework

Consider a scenario where the user interacts in conversation with a daily assistant. The user may raise queries to the assistant regarding their current needs to seek advice, and the assistant needs to provide appropriate solution suggestions for the user. Now, for each of the user's current needs under multiple topics, you are required to provide 8 appropriate solution suggestions for the user under each topic.

User Requirements
Below, the user's requirements and preferences for 1–3 conversation topics will be specified. "user_query" is the user's initial query, "implicit_needs" are the user's current implicit intents and relevant user background or experiences, and "requirement" is a summary of the above two parts, that is, the user's current actual need.
"""
<requirement>
"""

Specific Requirements and Output Template
Specific Requirements
1. Based on the content in the "User Requirements" section above, generate 8 appropriate solution suggestions for the user under each topic, with each solution described in one sentence.
2. For the 8 solutions under the same topic, while ensuring all solutions correspond to the user's requirements, they should be as diverse as possible, so that different solutions can be suitable for users with different preferences or backgrounds.
3. Output according to the JSON format template provided below, with the 8 solutions for each topic listed as an array.
4. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template to Complete (JSON format)
"""
<output_template>
"""

Now, following the above requirements and output template, please provide the generated result.

Step 5-3: Solution Preferences in Dialogue Framework

Consider a scenario where the user interacts in conversation with a daily assistant. The user may seek advice from the assistant on topics related to their own background or experiences. The assistant will provide some solution suggestions for the user, and the user may further give positive or negative feedback on the suggestions provided by the assistant. Now, given a user's identity background, personality, recent situation, current requirement topic, and corresponding preference description, as well as multiple candidate solution suggestions for the current requirement, please analyze and select the 2 solutions the user is most likely to like and the 2 they are most likely to dislike, and provide the corresponding reasons for the user's feedback.

User Background
"""
<background>
"""

User Personality
"""
<personality>
"""

User's Recent Situation
"""
<situation>
"""

User's Current Requirement
Below is information related to the user's current requirement. "user_query" is the user's initial query, "implicit_needs" are the user's current implicit intents and relevant background or experience, and "requirement" is a summary of the above two parts, i.e., the user's current actual need.
"""
<requirement>
"""

User Preference
Below is the user's preference description for similar requirements, divided into "pos" and "neg" parts, which respectively summarize the types of solutions the user likes and dislikes.
"""
<preference>
"""

Candidate Solution List
"""
<candidate_solutions>
"""

Specific Requirements and Output Template
Specific Requirements
1. The multiple candidate solutions listed above can be considered generally appropriate solutions for the current requirement, but they may not necessarily match the user's specific background or preferences. You need to analyze the user's attitude toward the above candidate solutions by combining the user's background and preference information provided in the input. Note: Solutions the user likes must match both the user's specific background and personalized preferences; if a solution fails to meet either, it should be considered a solution the user does not like.
2. Follow the JSON format output template below. First, provide an overall analysis of all candidate solutions under "analysis". Next, list the 2 solutions the user likes best and the 2 they like least in "pos_list" and "neg_list", respectively. For each solution, include the solution content ("solution") and the user's reason for positive or negative feedback ("feedback_reason").
3. The solution content ("solution") must be output exactly as in the corresponding candidate solution and cannot be modified in any way.
4. The user's feedback reason ("feedback_reason") should be output in the form of a single sentence, explaining from the user's perspective why they like or dislike the corresponding solution, citing their background or preferences.
5. Do not include the user's name in the output; use "the user" instead.
6. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template to Complete (JSON format)
"""
<output_template>
"""

Now, following the above requirements and output template, please provide the generated result.

Step 6-1: Logs

You are an excellent log generator. Consider a scenario where a user interacts with personal smart devices, and the device records various user interactions as logs. Now, given a user's basic information and a detailed description of the user's recent experiences, you are required to generate a series of corresponding logs based on the user's experiences.

<Interaction Log Types and Definitions>

Task Input
User Background
"""
<background>
"""

User's Recent Experiences (<time_span>)
"""
<experience>
"""

Specific Requirements and Output Format
Specific Requirements
1. You must follow the output format defined below. Each log entry includes four parts: timestamp ("timestamp"), event ("event"), log type ("type"), and log content ("content"). The "timestamp" must specify up to the minute, fall within the specified experience time range (<time_span>), and each subsequent log must have a later timestamp than the previous log. The "event" part should be a single sentence describing the user experience event reflected by this log. The format for log type ("type") and log content ("content") must follow the definitions in "Interaction Log Types and Definitions".
2. When generating logs, you need to comprehensively reflect all the details from the "User's Recent Experiences" input (try not to omit any event); the generated logs should be as dense as possible—each event can be reflected by multiple logs, and the total number of logs generated each time should be no less than 20.
3. The "User Background" information in the input is for reference only—just ensure there are no obvious contradictions between the generated logs and the user's background; you do not need to generate logs specifically targeting the background information.
4. For a given user experience event, logs can directly reflect the event (e.g., a purchase record on a medication platform can directly reflect the event "user bought medication"); they can also indirectly reflect the event (e.g., the user's browsing activity on a food delivery platform in the afternoon can indirectly reflect "user missed lunch at the cafeteria"). In addition, logs can appropriately supplement related behaviors before or after a particular experience event, such as adding logs about registration actions before a "user went to hospital for a checkup" event.
5. Ensure the rationality of logs. For example, when a user communicates online with others, the content can be directly reflected through the log types "message sent" and "message received"; but face-to-face communication should not be reflected by these two types, and you can consider using other log types to indirectly reflect such events, such as the user writing related communication content in a diary (not limited to this method).
6. When generating logs, you need to specify ambiguous time and location information from the user's experiences, e.g., "around 3 PM" can be converted to a specific moment within 15:00±15min, and a vague location like "in a hotel" can be converted to a hotel with a name and specific floor and room number. Note: you cannot use "XX" to replace specific location names; if needed, you can make up a concrete location name.
7. The platform and device type corresponding to the log should match the user's environment. For example, in mobile scenarios, mainly generate logs corresponding to common mobile device platforms (e.g., mini-programs, apps, etc.); in home environments, generate logs from smart home devices; while in office environments, you can generate logs from enterprise systems, web pages, etc., corresponding to PC devices. But ensure all logs originate from the user's personal devices, not public devices.
8. Ensure that each log entry reflects only one of the user's operation behaviors. If an event contains multiple operations, you need to synthesize multiple logs to represent the event.
9. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Format (JSON)
"""
[
    {
        "timestamp": "YYYY-MM-DD HH:MM", // Must be within the specified experience time range (<time_span>), and each subsequent log must have a later timestamp than the previous log.
        "event": "...", // A single sentence describing the user experience event reflected by this log
        "type": "...", // Must belong to a predefined log type
        "content": "..." // Generated log content must match the content format corresponding to the predefined type
    },
    ...
]
"""

Now, following the above requirements and output format, please generate the user's interaction logs.

Step 6-2: Dialogues

You are an excellent dialogue data generator. You can simulate appropriate user-assistant interaction dialogues according to my requirements.

Consider scenarios where a user asks an interactive assistant for daily advice. In the current situation, the user may face some daily needs and seek advice from the assistant; the assistant will try to understand the user's needs based on their background and experiences and provide corresponding solutions; and the user, according to their personal preferences, may give positive or negative feedback on the assistant's suggestions. You need to simulate the above user-assistant dialogue process, and indicate the dialogue action type for each utterance in the generated conversation.

<dialogue action definitions>

Task Input
I will provide you with a user's background, personality, current situation, dialogue content framework, and a predefined dialogue template. You must follow the dialogue content framework and dialogue template to generate specific user-assistant dialogue content.

User Background
""""""
<background>
""""""

User Personality
""""""
<personality>
""""""

User's Recent Situation
""""""
<situation>
""""""

Dialogue Framework
The following is the dialogue framework in JSON format, containing 1–3 interaction topics. The i-th topic is marked with the key "Ti". Each topic contains an interaction topic ("topic"), user requirements ("requirements"), and solution proposals ("solutions"). In each topic's user requirements, there is one initial user query (denoted as "Ti_Q") and 1–2 implicit user needs (the j-th is "Ti_Nj"); each implicit need contains a specific need description ("need") and a list of evidences from the user's background and experiences ("evidences"). Each topic's solution proposals contain 1–3 (the j-th is "Ti_Sj") solutions, and each solution ("solution") corresponds to a user feedback type ("feedback") and a feedback reason ("reason"). The user feedback types are positive ("pos") and negative ("neg"); "pos" suggestions align with the user's personalized preferences, "neg" suggestions do not, but both types are generally appropriate solutions to the current need. The feedback reason explains the user's positive or negative feedback for a solution.
""""""
<dialogue_framework>
""""""

Dialogue Template
The following dialogue template specifies the dialogue action ("action") and reference content ("reference") for each utterance in each round. The reference labels include {"Ti_Q", "Ti_Nj", "Ti_Sj"}, with meanings as explained in the "dialogue framework" section above.
""""""
<dialogue_template>
""""""

Specific Requirements and Output Format
Specific Requirements
1. You must strictly follow the dialogue template's specified dialogue action and reference content for each round.
2. When generating, assume the assistant role has some prior knowledge of the user's recent experiences, but does not know the user's overall background or preferences. The user role must follow the background, personality, situation, and preferences provided in the input.
3. When generating the user's "topic query" utterance, you may slightly adjust the content specified in the dialogue framework's "Ti_Q" according to the context, but do not change the main meaning.
4. When generating the assistant's "need inference" utterance, refer to the corresponding "Ti_Nj" in the dialogue framework. Use the "evidences" section with user background and experience to infer the implicit need described in "need" and seek confirmation from the user.
5. When generating the assistant's "solution proposal" utterance, you may slightly adjust the solution description from the corresponding "Ti_Sj" in the dialogue framework to better fit the user's background and situation, but you cannot entirely change the solution.
6. When generating the user's "solution feedback" utterance, refer to the "feedback" type and "reason" in the corresponding "Ti_Sj" in

the dialogue framework, and briefly, directly express the user's request or attitude. Especially when facing an inappropriate suggestion ("feedback" is "neg"), the user should directly express negative attitude, even anger or dissatisfaction. For inappropriate solutions, avoid polite phrases like "sounds good, but..." or "looks nice, but...".

7. The dialogue should flow as smoothly as possible between utterances. For example, if the user's action in a round is "solution feedback" for the previous solution and the assistant's action is "solution proposal" for a new solution, the assistant's reply may briefly acknowledge the user's feedback before recommending a new solution. Also, different topics should transition smoothly; for example, when generating the user's "topic query" for the second topic, you may include some connecting words or phrases from the first topic to the current one.

8. Do not introduce solutions not specified in the "dialogue framework", especially after negative user feedback; if the assistant's action is "feedback response" rather than "solution proposal", simply express apology or regret or ask if the user has other issues, without trying to provide extra suggestions.

9. Do not mention the user's name in the dialogue unless necessary.

10. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Format (JSON)
"""
```
{
    "turn_1": {
        "user": {"action": "...", "reference": "...", "content": "..."},
        "assistant": {"action": "...", "reference": "...", "content": "..."}
    },
    ...
}
```
"""

Now, according to the above requirements and output format, please provide the generated user-assistant dialogue.

Prompts of PAL-Bench Tasks

Task 1: Requirement Restatement

You are a personalized interactive assistant, able to understand user needs by combining user history.

You are interacting with the user. Now you are given some user personalized information as reference, as well as the user's current query. Please use the user's history to deeply understand the user's actual current need, and describe the user's actual need in one sentence.

User Personalized Information
<memory>

Current User Query
"""
<user_query>
"""

Output Template
Below is the output template in JSON format, where "requirement" is the content to be generated.
"""
{"requirement": "..."}
"""

Output Requirements
1. Combine the user's historical information provided in the input to understand the user background and context related to the current query, and describe the user's actual need in one sentence.
2. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Now, according to the output requirements and template, please provide your output.

GPT-Score Evaluation of Task 1

You are a good scorer who can score the outputs generated by the assistant model according to requirements.

Consider a scenario where a user interacts with a daily assistant. The assistant model needs to combine the user's personalized information to deeply understand the user's implicit needs for the current inquiry and provide a description of the user's actual need.

Now, you are given the user's initial question as background, as well as the candidate model's prediction and the reference content of the user's actual need as the prediction target. You are required to compare the prediction to the reference and score the model's prediction performance.

User Initial Query
"""
<user_query>
"""

User Actual Requirement (Reference)
Contains two parts: the complete user requirement description ("requirement") and a list of the user's implicit needs ("implicit_needs").
The implicit needs list contains 2 entries, corresponding to two aspects of the user's implicit needs not explicitly mentioned in the inquiry.
"""
<reference>
"""

Candidate Model Prediction
"""
<prediction>
"""

Evaluation Requirements and Output Template
Evaluation Requirements
1. The provided "User Initial Query" in the input is only for background; please focus on the match between the prediction and the reference.
2. In the input "User Actual Need" (the reference), the complete need description ("requirement") can be regarded as a summary that combines the two entries in the implicit needs list ("implicit_needs"), and can serve as a reference for the prediction. However, your evaluation should focus on the degree to which the prediction matches the two entries in the implicit needs list.
3. The scoring range is in [0, 2]. If the prediction matches both entries in the reference, it scores 2 points; if the prediction only matches one entry, it scores 1 point; if the prediction matches neither, it scores 0 points. The order of the two reference entries does not matter.
4. When considering the match between the prediction and the reference implicit needs entries, focus on whether the core elements of each reference entry are reflected in the prediction; do not pay too much attention to exact wording. If a part of the prediction provides more detailed information in the same aspect as a reference entry, it is considered a match.
5. Considering that the prediction and a reference entry may be partially matched, you may assign 0.5 points for a partially matched entry.
6. Output in the following JSON format: Provide an analysis paragraph for the current evaluation, followed by the score for the candidate model.

Output Template
"""
{
"analysis": "...", // Evaluation analysis
"score": <score>// Value range: {0, 0.5, 1, 1.5, 2}
}
"""

Now, please provide your evaluation analysis and score as required.

Task 2-1: Solution Generation

You are a personalized assistant, able to propose solutions that match the user's personalized preferences according to their needs.

You are interacting with the user. Now you are given some user personalized information as reference, as well as the user's recent logs and the current user requirement description. Please use the user's history to deeply understand the user's personalized preferences, and provide a solution that matches their preferences for the current user requirement.

User Personalized Information
<memory>

Current User Requirement
"""
<requirement>
"""

Output Template
Below is the output template in JSON format, where "solution" is the content to be generated.
"""
{"solution": "..."}
"""

Output Requirements
1. Combine the user's historical information provided in the input to understand the user's personalized preferences, and provide a solution that matches the user's preferences. The solution should be described in one sentence in the output.
2. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Now, according to the output requirements and template, please provide your output.

Task 2-2: Solution Selection

You are a personalized assistant, able to provide solutions that match the user's personalized preferences according to their needs.

You are interacting with the user. Now you are given some user personalized information as reference, as well as the current user requirement description and 8 candidate solution suggestions for the current requirement. Please use the user's history to deeply understand the user's personalized preferences and select from the candidate solutions the 2 that best match the user's preferences.

User Personalized Information
<memory>

Current User Requirement
"""
<requirement>
"""

Candidate Solution Suggestions
Below are 8 candidate solutions for the current user requirement, including the id and content for each solution.
"""
<candidate_solutions>
"""

Output Template
Below is the output template in JSON format, where "solution" is the content to be generated.
"""
{
"analysis": "...", // Provide an overall analysis of all candidate solutions, focusing on the user's personalized preferences
"selected_solutions": [...] // List the ids of the 2 solutions that best match the user's preferences
}
"""

Output Requirements
1. Combine the user's historical information provided in the input to understand the user's personalized preferences, and analyze and select candidate solutions based on the user's preferences.
2. Follow the output template above. In the "selected_solutions" section, only the 2 solution ids that best match the user's preferences may be selected—no more, no fewer.
3. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Now, according to the output requirements and template, please provide your output.

Task 3: User-LLM

You are an excellent user simulator. Consider a scenario where a user interacts in dialogue with an assistant, seeking advice about topics related to themselves and hoping the assistant can accurately infer their needs and provide solutions that match their preferences. You need to simulate this user according to the personalized information provided in the input, and generate user utterances of the specified type in each round of the conversation. The user utterance type is one of {<topic query>, <need confirmation>, <solution discussion>, <solution feedback>}.

Now, you are given some personalized information about this user, the current interaction context, and the type of user utterance you (as the user) need to generate next. Please generate the corresponding user utterance content according to the user's personalized

characteristics and the specified utterance type.

User Personalized Information
<persona>

Current Interaction Status
Current Dialogue Context
Below is the dialogue context between you ("user") and the assistant ("assistant"). The last user utterance is the one you need to generate, and its utterance type has been specified.
"""
<dialogue_context>
"""

Current Dialogue Turn
"""
<current_turn>
"""

Current User Utterance Type Definition (<action>)
"""
<action_description>
"""

Output Template and Output Requirements
Output Requirements
1. Combine the information provided in the input and follow the specified user utterance type to generate an appropriate interaction utterance as the user ("user").
2. You must follow the utterance type specified in the current dialogue turn and the definition of this type provided in the input.
3. While following the specified utterance type, you may include some transitional phrases in the generated content to ensure smoother context flow.
4. The user utterance should be a relatively brief sentence.
5. The user should be able to simply and directly express their attitude, rather than always overly accommodating the assistant's reply. If the assistant's inference of the user's need is off, or the proposed solution does not match the user's preferences, the user should directly express negative attitude, even anger or dissatisfaction; avoid overly euphemistic expressions such as "sounds great, but..." or "looks nice, but...".
6. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
Below is the output template in JSON format, where "content" is the user utterance to be generated.
"""
{"content": "..."}
"""

Now, according to the output requirements and template, please provide your output.

Task 3: Assistant-LLM

You are a personalized dialogue assistant who can generate dialogue replies that meet the user's personalized needs and preferences according to the specified reply type. Each utterance's reply type is one of {<need inference>, <solution proposal>, <solution discussion>, <feedback response>}.

You are currently interacting in dialogue with the user. Now you are given some user personalized information as a reference, as well as the dialogue context between you and the user. Please provide an appropriate dialogue reply according to the specified reply type.

User Personalized Information
<memory>

Current Dialogue Context
Below is the current dialogue context between the user ("user") and you ("assistant"). The last assistant utterance is the one you need to generate, and its reply type has been specified.
"""
<dialogue_context>
"""

Current Dialogue Turn
"""
<current_turn>
"""

Current Reply Type Definition (<action>)
"""
<action_description>
"""

Output Template
Below is the output template in JSON format, where "content" is the reply content to be generated.
"""
{"content": "..."}
"""

Output Requirements
1. Combine the information provided in the input and follow the specified reply type to generate an appropriate dialogue reply as the assistant ("assistant").
2. You must follow the reply type specified in the current dialogue turn and the definition of this type provided in the input.
3. While following the specified reply type, you may include some transitional phrases in the reply to make the context flow more smoothly. For example, for a "solution proposal" type reply, you may briefly respond to the user's feedback on the previous solution (if any) before recommending a new solution.
4. The reply content does not need to be long—1 to 3 sentences is sufficient.
5. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Now, according to the output requirements and template, please provide your output.

Evaluation for Task 3 on the Requirement Dimension

You are an excellent dialogue evaluator. Consider a scenario where a user interacts in dialogue with an assistant, seeking advice about topics related to themselves. The assistant first tries to infer the user's needs and then attempts to provide solutions that match the user's preferences. Now you are given two separate conversations between the same user and two different assistants on the same topic. You need to compare and evaluate the assistants' replies according to my instructions.

You are given some personalized information about the user ("user"), including the user's background, personality, recent situation, and detailed information about the user's current need. Based on the user's personalized information, you are to judge how well the two assistants ("assistant-1"/"assistant-2") understood and inferred the user's needs during their respective interactions.

User Personalized Information
User Background
"""
<background>
"""

User Personality
"""
<personality>
"""

User's Recent Situation
"""
<situation>
"""

User's Current Requirement
Below is information related to the user's current requirement. "user_query" is the user's initial content when actively asking the assistant; "implicit_needs" are the user's current implicit needs and related background or experiences, which the user expects the assistant to proactively infer in the conversation; "requirement" is a summary of the above two parts, i.e., the user's current actual need.
"""
<requirement>
"""

Dialogue Content to Evaluate
Below are the conversations between the user ("user") and two assistants ("assistant-1"/"assistant-2"). You should focus on the effectiveness of the assistants' replies.

assistant-1
""""
<dialogue_assistant_1>
""""

assistant-2
""""
<dialogue_assistant_2>
""""

Current Evaluation Dimension: Need Understanding
In each conversation between the user and the assistant, the dialogue revolves around a user need (i.e., the "requirement" content in the "User's Current Requirement" section). The assistant and user interact through the assistant's inference of needs and solution proposals. In this evaluation, focus on the assistant's "need understanding" ability, as described below:
1. You should focus on the "User's Current Requirement" section in the user's personalized information and judge whether the assistant proactively inferred the user's implicit needs and corresponding situational background information that the user did not mention.
2. The more specific and accurate the assistant's proactive inference, the higher the score should be; conversely, if the inference is vague or inaccurate, a lower score should be given.
3. In the conversation, the user may clarify some need content. After the user clarifies, if the assistant merely repeats the relevant content, this should not be considered a plus. Points should be given for the assistant's proactive inference of content the user has not mentioned. Also, in multi-turn interactions, the assistant who gives more specific and accurate need inferences earlier should be considered for a higher score.
4. This evaluation should focus mainly on the assistant's inference of the user's needs in the dialogue, and evaluate based on how well the assistant understood the user's needs. There is no need to consider the assistant's solution proposals and discussions.

Evaluation Requirements and Output Template
Evaluation Requirements
1. Combine the user's personalized information and the dialogues between the user and the two assistants, and provide your output using the JSON format output template below. First, provide an analysis ("analysis") of the two assistants' reply effectiveness, then assign scores ("scores") to both assistants, with scores ranging from 1 to 10 (integer values).
2. In the "analysis" section, first analyze the dialogue reply effectiveness of both assistants separately, then provide an overall analysis and comparison. Note: The assistants' names ("assistant-1"/"assistant-2") are randomly assigned to the two specific assistants, so the order of appearance should be considered completely unrelated to the reply effectiveness. Please ensure the assistants' relative order does not affect your evaluation.
3. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
""""
{
    "analysis": { // First analyze both assistants' reply effectiveness separately, then provide an overall analysis and comparison
        "assistant-1": "...",
        "assistant-2": "...",
        "overall": "..."
    },
    "scores": { // Score range: integers from 1 to 10
        "assistant-1": <score>,
        "assistant-2": <score>
    }
}
""""

Now, according to the evaluation requirements and template, please provide your evaluation output.

Evaluation for Task 3 on the Preference Dimension

You are an excellent dialogue evaluator. Consider a scenario where a user interacts in dialogue with an assistant, seeking advice about topics related to themselves. The assistant first tries to infer the user's requirements and then attempts to provide solutions that match the user's preferences. Now you are given two separate conversations between the same user and two different assistants on the same topic. You need to compare and evaluate the assistants' replies according to my instructions.

You are given some personalized information about the user ("user"), including the user's background, personality, recent situation, and the user's current requirement and related preferences. Based on the user's personalized information, you are to judge how well the two assistants ("assistant-1"/"assistant-2") understood the user's preferences when proposing solutions to the user's requirements during their respective interactions.

User Personalized Information
User Background
"""
<background>
"""

User Personality
"""
<personality>
"""

User's Recent Situation
"""
<situation>
"""

User's Current Requirement and Related Preferences
In the following information, "requirement" is a detailed description of the user's current requirement; "general_preference" gives the user's overall preference description for this type of requirement, divided into "pos" and "neg" parts, summarizing the types of solutions the user likes and dislikes, respectively; in "candidate_solutions", "pos_list" and "neg_list" list 2 specific solutions each that match or do not match the user's preferences for the current requirement.
"""
<preference>
"""

Dialogue Content to Evaluate
Below are the conversations between the user ("user") and two assistants ("assistant-1"/"assistant-2"). You should focus on the effectiveness of the assistants' replies.

assistant-1
"""
<dialogue_assistant_1>
"""

assistant-2
"""
<dialogue_assistant_2>
"""

Current Evaluation Dimension: Preference Understanding
In each conversation between the user and the assistant, the dialogue revolves around a user requirement (i.e., the "requirement" content in the "User's Current Requirement and Related Preferences" section). The assistant and user interact through the assistant's inference of requirements and solution proposals. In this evaluation, focus on the assistant's "preference understanding" ability, as described below:
1. You should focus on the "User's Current Requirement and Related Preferences" section in the user's personalized information, understand the user's preferences for different types of solutions under the current requirement, and judge whether the solutions proposed by the assistant in the dialogue match the user's personalized preferences.
2. If the assistant's proposed solutions highly match the user's positive preferences, they should receive a higher score; conversely, if the solutions do not match the user's positive preferences, or even align more with the user's negative preferences (i.e., the types of solutions the user dislikes), they should receive a lower score.
3. In the dialogue, the user may provide feedback on some solutions and may proactively state or reveal specific preferences. When evaluating the assistant's preference understanding ability, give higher priority to cases where the assistant proposes solutions that match the user's preferences before the user reveals those preferences.
4. This evaluation should focus mainly on the assistant's solution suggestions for the user's requirements, and evaluate based on how well the solutions match the user's preferences. There is no need to consider the assistant's inference and confirmation of the user's requirements.

Evaluation Requirements and Output Template
Evaluation Requirements
1. Combine the user's personalized information and the dialogues between the user and the two assistants, and provide your output using

the JSON format output template below. First, provide an analysis ("analysis") of the two assistants' reply effectiveness, then assign scores ("scores") to both assistants, with scores ranging from 1 to 10 (integer values).

2. In the "analysis" section, first analyze the dialogue reply effectiveness of both assistants separately, then provide an overall analysis and comparison. Note: The assistants' names ("assistant-1"/"assistant-2") are randomly assigned to the two specific assistants, so the order of appearance should be considered completely unrelated to the reply effectiveness. Please ensure the assistants' relative order does not affect your evaluation.

3. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
{
    "analysis": { // First analyze both assistants' reply effectiveness separately, then provide an overall analysis and comparison
        "assistant-1": "...",
        "assistant-2": "...",
        "overall": "..."
    },
    "scores": { // Score range: integers from 1 to 10
        "assistant-1": <score>,
        "assistant-2": <score>
    }
}
"""

Now, according to the evaluation requirements and template, please provide your evaluation output.

Prompts of H^{2}Memory Method

Step 1-1 of Memory Construction: Log Graph

You are given some user logs, each containing a timestamp and specific log content. For each log within a specified range, please generate a list of its related previous logs. Specifically, suppose the current log corresponds to <Event B>, and a previous log corresponds to <Event A>. Consider two types of relationships:

1. "caused_by": <Event A>is the direct cause of <Event B>. If <Event A>did not occur, <Event B>would not occur.
2. "follows": <Event B>and <Event A>belong to the same topic and have a sequential relationship in time, i.e., <Event B>temporally follows <Event A>. But <Event B>'s occurrence does not entirely depend on <Event A>.

Logs
"""
<logs>
"""

Detailed Requirements and Output Template
Detailed Requirements
For each log within the input (<logs_id_span>), consider its relationships with previous logs and list the previous log ids for both "caused_by" and "follows" types.
Note:

1. You need to iterate over all logs in the specified range and generate the output results. For any given log, only previous logs can be in its relationships (related logs can include logs before the specified range).
2. Only consider clear, direct relationships—no need to consider indirect relationships. For example: if "log_0" causes "log_2", and "log_2" causes "log_4", you do not need to consider "log_0" as causing "log_4".
3. You must consider the two relationship types in the order "caused_by", then "follows". If two logs are already considered to have a "caused_by" relationship, do not consider a "follows" relationship between them. Only consider "follows" if there is no "caused_by" relationship between the two logs.
4. Imitate the output example below and output in JSON format. Do not output anything except the required JSON; do not add any extra explanations or comments in the JSON.

Output Template
"""
{
    "<start_log_id>": { // Current log
        "caused_by": [...], // List of previous log ids for this relationship type; leave empty if none
        "follows": [...]
    },

```
    ... // Iterate over all logs in the specified range (<logs_id_span>)
}
"""
```

Now, according to the above requirements and format, please provide the related log lists for each log.

Step 1-2 of Memory Construction: Situation

You are given some user logs, each including a timestamp, specific log content, and the relationships between different logs. Please use the provided relationships to connect the content of each log, first generating a situation description that infers and summarizes the user's recent experiences or events; then, determine which one or more of the following four aspects the situation primarily belongs to: work, health, family, leisure.

Relationship Definitions
Suppose the current log corresponds to <Event B>and a previous log corresponds to <Event A>. Two possible relationship types are defined:
1. "caused_by": <Event A>is the direct cause of <Event B>. If <Event A>did not occur, <Event B>would not occur.
2. "follows": <Event B>and <Event A>belong to the same topic and have a sequential relationship in time, i.e., <Event B>temporally follows <Event A>. But the occurrence of <Event B>does not entirely depend on <Event A>.

Logs
"""
<subgraph>
"""

Detailed Requirements and Output Template
Detailed Requirements
1. Follow the JSON output template below. "situation" is the situation description generated by connecting the logs, and "situation_aspects" are the aspects the situation belongs to.
2. The "situation" description should be a concise paragraph summarizing the user's experiences or events during the time period corresponding to the logs. Since user logs mostly reflect the user's experiences or events indirectly, the situation description does not need to exactly quote the descriptions from the logs, but can directly describe the experiences or events reflected by the logs. In addition, unless necessary, there is no need to include specific time points in the description; if there are multiple logs, just reflect the relationships between the logs.
3. The value of "situation_aspects" should be in Python list format, with possible values {"work", "health", "family", "leisure"}, representing the main aspect(s) of the current situation. The list can contain one or more elements, but cannot be empty.
4. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
{
    "situation": "...", // Situation description in paragraph form
    "situation_aspects": [...] // List of aspects the situation belongs to
}
"""

Now, according to the above requirements and output format, please generate the user's situation.

Step 2-1 of Memory Construction: Topic Outline

Consider a scenario where a user ("user") interacts in dialogue with a daily assistant ("assistant"). The user may ask the interactive assistant for advice on topics related to their background or experiences; the assistant, after fully understanding the user's needs, provides solution suggestions; and the user may further give feedback on the solutions proposed by the assistant. Now, you are given a conversation between the user and the assistant. Please summarize the outline information for each topic in the conversation, including the dialogue turn range for the topic, the user's need for that topic, the solutions provided by the assistant, the user's feedback on each solution, and the user's preferences as reflected in the above content.

Dialogue
"""
<dialogue>
"""

Detailed Requirements and Output Template

Detailed Requirements

For the given conversation, you need to summarize the outline information for each topic according to the output template below, including the dialogue turn range for the topic ("turn_span"), the user's need for that topic ("requirement"), the solutions provided by the assistant ("solution_list"), and the user's preferences as reflected in the interaction for that topic ("preference").

Specific requirements and explanations are as follows:

1. There may be multiple topics in a conversation, each revolving around a user need. The interaction includes understanding the user's need and discussing solutions for the current need, possibly spanning multiple consecutive dialogue turns. The dialogue turn range ("turn_span") for each topic is a list of two elements, representing the turn id where the topic starts and the turn id where it ends (turn id format is "turn_x"). The turn ranges of different topics in the conversation must be contiguous.

2. The user need ("requirement") should be described in one sentence, describing the user's concrete intention for advice and need content in the current context and based on their background.

3. The "solution_list" section should enumerate one or more solutions discussed by the user and assistant under the current topic. For each solution, you must provide the solution content ("solution"), a description of the user's feedback ("user_feedback"), and the feedback type ("feedback_type"). The solution content should summarize the solution discussed in one sentence; the user feedback should briefly describe the user's attitude toward the corresponding solution (from the assistant's perspective); and the feedback type should be classified as positive ("pos"), negative ("neg"), or other ("others"). Only use "others" if the user's attitude is truly unclear; otherwise, prefer "pos" or "neg".

4. In the conversation, discussion about each solution may span several sentences or turns; the "solution" section should provide a summary description for each solution, and the "feedback_type" section should focus on the user's final attitude toward the solution after discussing it with the assistant. Prefer classifying feedback as "pos" or "neg"—only use "others" if the user's final attitude is very ambiguous.

5. The "preference" section should focus on the user's preferences as reflected by their feedback on different solutions for the current topic, and summarize the user's preferences for this particular need in one sentence. Note: If there is negative user feedback on a certain aspect in the dialogue, you should reflect the user's negative preferences (i.e., solution types the user may dislike or tends to reject) in the summary.

6. Follow the output template below and output in JSON format.

7. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
[
    {
        "turn_span": [<start_turn_id>, <end_turn_id>], // Dialogue turn range, turn id format is "turn_x"
        "requirement": "...", // User need description
        "solution_list": [
            {
                "solution": "...", // Solution proposed by the assistant
                "user_feedback": "...", // User's feedback attitude toward the solution
                "feedback_type": "..." // Feedback type, possible values: {"pos", "neg", "others"}; prefer "pos" or "neg"
            },
            ...
        ],
        "preference": "..." // User's preferences for the current need as reflected in the interaction
    },
    ...
]
"""

Now, according to the above requirements and format, please generate topic outlines for the current conversation.

Step 2-2 of Memory Construction: Requirement Refinement

Consider a scenario where a user ("user") interacts in dialogue with a daily assistant ("assistant"). The user may ask the interactive assistant for advice on topics related to their background or experiences, thus starting an interaction. Now, suppose we have already summarized the user's requirement based on the dialogue content. You are required to rewrite the original user requirement as a more detailed description, using the user's background and recent situation provided in the input, to achieve a deeper understanding of the user's requirement.

User Overall Background
"""
<background>
"""

User's Recent Situation
"""
<situation>
"""

User Requirement
"""
<requirement>
"""

Detailed Requirements and Output Template
Detailed Requirements
1. The user's overall background and recent situation provided in the input may contain reasons or specific events that led to the current user requirement, as well as unrelated events or information. You should use the background and situation factors relevant to the current requirement to rewrite the provided "User Requirement" as a more detailed description.
2. The rewritten content should still be a single sentence and should remain focused on one core requirement, without introducing unrelated requirements or background factors not relevant to the current requirement.
3. Follow the output template below and respond in JSON format.
4. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.
Output Template
"""
{"requirement": "..."}
"""

Now, according to the above requirements and format, please rewrite the currently provided requirement content.

Step 3 of Memory Construction: Background Summary

Consider the problem of summarizing and updating user background information based on long-term user history. Now, you are given a previous summary of the user's background, as well as some recent specific situations. You need to consider updating the user background according to the content of these recent situations.

Previous User Background Summary
The following is the previous summary of the user's background in four aspects ("background"), as well as the corresponding time span ("time_span"). The four aspects of the user's background are work, health, family, and leisure, each summarized in one sentence. If a section is empty, it means that no valid information for that aspect was previously included in the user's history.
"""
<last_background>
"""

List of Recent User Situations
The following is a list of recent user situations that have not yet been updated into the user background. The time span is "time_span", and "situation_list" lists all recent situations.
"""
<cur_situations>
"""

Detailed Requirements and Output Example
Detailed Requirements
For the given list of recent situations, you should first determine which aspects among work, health, family, and leisure in the user background need to be updated, and then, for those aspects that need updating, provide the updated background summary.
Note:
1. The user background should be a summary describing the user's overall characteristics or attributes in the corresponding aspect, such as occupation, health status, family members, leisure hobbies, etc. You do not need to include overly specific events in the user background.
2. If the given situations do not contain information about a certain aspect, do not update the background for that aspect; even if the situations contain information about an aspect, if you believe that aspect is already reflected in the background summary or is too specific to include in the overall background, you should also keep the original background summary unchanged.
3. The updated background summary for each aspect should still be a one-sentence summary. Keep sentences as concise as possible; avoid repetition. When updating, you should consider compressing both the previous background and the new information as needed, and, if necessary, reorganize the original summary sentence to prevent the sentence from growing too long.
4. The time span of the previous background summary and the recent situations is only for reference when updating; if the previous background summary covers a long time period, try to minimize the impact of recent situation information on the background. There is no need to mention specific time in the updated background summary.

5. Imitate the JSON output example below: first list the aspects of the user background that need updating ("updating_aspects"), then provide the updated summary for each aspect ("updating_content").
6. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Example
"""
{
    "updating_aspects": ["xxx"], // Aspects you believe need updating based on recent situations
    "updating_content": {
        "xxx": "..." // Provide the updated summary for aspect xxx
    }
}
"""

Now, according to the above requirements and format, please provide the updated user background information based on the current situations.

Step 4-1 of Memory Construction: Requirement Abstraction

Consider a scenario where a user interacts with a daily assistant. The assistant has already summarized a series of user requirements in specific contexts based on their interaction history. Now, you are given a batch of requirement contents that are considered semantically similar, and you need to generalize this batch of specific requirements into a more abstract overall requirement type description.

Specific Requirement Contents
The input is a list, each entry in the list is a specific user requirement:
"""
<requirements>
"""

Detailed Requirements and Output Template
Detailed Requirements
Please generalize all of the above given specific user requirements into a short phrase describing the user's overall requirement type.
Note:
1. The output should be a concise short phrase, without including situational details from the specific requirement contents.
2. The multiple specific requirements given in the list should be considered to belong to the same macro aspect, and the overall requirement type description should reflect this macro aspect.
3. When the number of specific requirement contents is small (such as just one), the generalized overall requirement type should still be an abstraction of the original requirement content.
4. Imitate the output example below and output in JSON format.
5. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
{"general_requirement": "..."} // A short phrase describing the user's overall requirement type
"""

Now, according to the above requirements and format, please provide your output.

Step 4-2 of Memory Construction: Preference Abstraction

Consider a scenario where a user interacts with a daily assistant. The assistant has already summarized a series of specific user preference experiences under a certain requirement type based on their interaction history. Now, you are given the specified requirement type and multiple corresponding user preference experiences. You are asked to further generalize these experiences into a more abstract and broadly applicable principle of preference.

Requirement Type
"""
<general_requirement>
"""

User Preference Experiences
The input is a list, each entry in the list is a user preference experience:
"""

"""

Detailed Requirements and Output Template
Detailed Requirements
Please generalize all the above experiences into a one-sentence principle of user preference.
Note:
1. The output principle should be a concise one-sentence form.
2. The multiple specific preference experiences given in the list should all be considered as corresponding to the "requirement type" given above, and the output principle should reflect the overall preference under that requirement type.
3. When the number of preference experiences is small (such as just one), the generalized principle should still be an abstraction of the original experience.
4. Imitate the output example below and output in JSON format.
5. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
{"principle": "..."} // A one-sentence user preference principle
"""

Now, according to the above requirements and format, please provide your output.

Step 4-3 of Memory Construction: Requirement Update

Consider a scenario where a user interacts with a daily assistant. The assistant summarizes a series of user requirements in specific contexts based on the interaction process, and also generalizes multiple requirements of the same type into a more abstract overall requirement type description. Now, suppose the assistant has already generalized multiple previous requirements of the same type into an overall requirement type, but there are now some new specific requirements of that type. You need to determine whether the previous overall requirement type description needs to be adjusted, and if so, provide the revised description.

User Overall Requirement Type
"""
<general_requirement>
"""

Newly Added Specific Requirement Contents
The input is a list; each entry in the list is a newly added specific user requirement:
"""
<requirements>
"""

Detailed Requirements and Output Template
Detailed Requirements
Please determine whether the newly added requirement contents already belong to the given overall requirement type. If the new requirements conflict with the given overall requirement type, you need to update the overall requirement type description.
Note:
1. The existing overall requirement type description is already a generalization of similar requirements in the interaction history, so you should keep the original overall requirement type description unchanged unless necessary.
2. If you need to revise the original overall requirement type description, you should comprehensively consider both the original description and the new requirement contents. The updated overall requirement type description should still meet the following requirements: it should be a concise short phrase, not include specific details (especially not concrete examples), and describe the overall requirement type at a more abstract macro level; there is no need to include user preference information.
3. Imitate the output examples below, and output in JSON format whether the original overall requirement type needs adjustment, and if so, the updated description.
4. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
If you believe the previous overall requirement type does not need adjustment, output:
"""
{
    "adjust": "No"
}
"""

If you believe the previous overall requirement type needs adjustment, output the judgment and the revised description:
"""
{
    "adjust": "Yes",
    "general_requirement": "..." // The revised overall requirement type description, still in the form of a short phrase
}
"""

Now, according to the above requirements and format, please provide your output.

Step 4-4 of Memory Construction: Preference Update

Consider a scenario where a user interacts with a daily assistant. The assistant summarizes a series of specific user preference experiences under a certain requirement type based on the interaction process, and further generalizes multiple preference experiences into a more abstract and broadly applicable user preference principle. Now, suppose the assistant has already generalized multiple previous preference experiences under a certain requirement type into a principle, but there are now some newly added preference experiences of that type. You need to determine whether the previous user preference principle needs to be adjusted, and if so, provide the revised principle.

User Requirement Type
"""
<general_requirement>
"""

User Preference Principle
"""
<principle>
"""

Newly Added Preference Experiences
The input is a list; each entry in the list is a newly added preference experience description:
"""
<preferences>
"""

Detailed Requirements and Output Template
Detailed Requirements
Please determine whether the newly added preference experience descriptions already conform to the given user preference principle. If the new preference experiences conflict with the given principle, or if the existing principle obviously omits some important content reflected in the current experiences, you need to update the user preference principle.
Note:
1. The existing user preference principle is already a generalization of similar preference experiences in the interaction history, so you should keep the original user preference principle unchanged unless necessary.
2. If you need to revise the original principle, you should comprehensively consider both the original principle and the new experiences. The updated user preference principle should still meet the following requirements: it should be a concise one-sentence form, a further abstraction and generalization of the specific experiences, and it should correspond to the "requirement type" provided in the input.
3. Imitate the output examples below, and output in JSON format whether the principle needs adjustment, and if so, the updated content.
4. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
If you believe the previous user preference principle does not need adjustment, output:
"""
{
    "adjust": "No"
}
"""
If you believe the previous user preference principle needs adjustment, output the judgment and the revised principle:
"""
{
    "adjust": "Yes",
    "principle": "..." // The revised user preference principle, still in the form of a concise one-sentence statement
}
"""

Now, according to the above requirements and format, please provide your output.

Query Generation for Multi-turn Dialogue Interaction

You are interacting with a user in dialogue, and the interaction process may include the user's needs as well as discussions of solutions to the current need. The input will provide the dialogue context between you and the user. You are required to summarize the user's needs as expressed in the conversation.

Dialogue Context
Below is the current dialogue context between the user ("user") and you ("assistant").
"""
<dialogue_context>
"""

Output Requirements and Output Template
Output Requirements
1. Imitate the output template below and provide your output in JSON format. The user's need ("requirement") should be summarized in a concise one-sentence form, describing the specific need for advice that the user wants to address in the interaction.
2. In the need description, you do not need to pay attention to the user's preferences regarding solutions.
3. Do not output any content other than the required JSON format. Do not add any extra explanations or comments in the JSON.

Output Template
"""
{
    "requirement": "..." // The user's need for the current topic, summarized in a concise sentence
}
"""

Now, please provide your output according to the output requirements and template above.

References

Ding, N.; Chen, Y.; Xu, B.; Qin, Y.; Hu, S.; Liu, Z.; Sun, M.; and Zhou, B. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 3029–3051.

Fleiss, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5): 378.

Jang, J.; Boo, M.; and Kim, H. 2023. Conversation Chronicles: Towards Diverse Temporal and Relational Dynamics in Multi-Session Conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 13584–13606.

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 (Volume 1: Long Papers)*, 13851–13870.

McCrae, R. R.; and John, O. P. 1992. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2): 175–215.

McHugh, M. L. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3): 276–282.

Wang, P.; Li, L.; Chen, L.; Cai, Z.; Zhu, D.; Lin, B.; Cao, Y.; Kong, L.; Liu, Q.; Liu, T.; et al. 2024. Large Language Models are not Fair Evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 9440–9450.

Wang, Q.; Fu, Y.; Cao, Y.; Wang, S.; Tian, Z.; and Ding, L. 2025. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 130193.

Wu, D.; Wang, H.; Yu, W.; Zhang, Y.; Chang, K.-W.; and Yu, D. 2025. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *The Thirteenth International Conference on Learning Representations*.

Xu, J.; Szlam, A.; and Weston, J. 2022. Beyond Goldfish Memory: Long-Term Open-Domain Conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5180–5197.

Yuan, R.; Sun, S.; Li, Y.; Wang, Z.; Cao, Z.; and Li, W. 2025. Personalized Large Language Model Assistant with Evolving Conditional Memory. In *Proceedings of the 31st International Conference on Computational Linguistics*, 3764–3777.

Zhang, S.; Dinan, E.; Urbanek, J.; Szlam, A.; Kiela, D.; and Weston, J. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2204–2213.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623.

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.