Behavior2Trip: Towards Personalized Travel Planning via User Behavior Trajectory

Zihao Cheng[1*], Yingyu Shan[2*], Hongru Wang[3*], Zeming Liu[1†], Xinyi Wang[4], Xiangrong Zhu[4‡], Yuhang Guo[2], Wei Lin[4], Yunhong Wang[1]

[1]School of Computer Science and Engineering, Beihang University, Beijing
[2]Beijing Institute of Technology [3]University of Edinburgh [4]Independent Researcher
[*]Equal Contribution [†]Corresponding Author [‡]Project Leader
Email: zihaocheng@buaa.edu.cn, zmliu@buaa.edu.cn

Abstract

Travel planning agents assist users in generating personalized travel plans by modeling their individual preferences. Existing agents either rely on explicit user instructions or engage in multi-turn clarification to elicit user preferences. However, both approaches overlook the rich behavioral signals latent in users' past behaviors, which implicitly encode their preferences. This over-reliance on active user input increases interaction burden and limits plan personalization. To bridge this gap, we introduce a new task, **Behavior-Aware Travel Planning**, which infers user preferences directly from past behaviors and generates personalized travel plans. To facilitate research on this task, we introduce **Behavior2Trip**, a benchmark constructed from one of the largest Chinese online travel platforms, comprising 11,400 instances. Each instance represents an average of 39.8 past user behaviors spanning 14 attributes across 5 preference dimensions. We further propose **B2T-Agent**, a reinforcement learning-based agent that leverages user behavior trajectories, interacts with external tools for preference-aligned retrieval, and maintains an internal memory module. Experiments on Behavior2Trip show that GPT-4.1 achieves a full-constraint pass rate of only 0.5% on the hardest tasks, while B2T-Agent built upon Qwen3-8B outperforms all baselines, highlighting the substantial challenge of this task. Moreover, Qwen3-8B trained with B2T-Agent also outperforms GPT-4.1 on the TravelPlanner benchmark, demonstrating strong generalization[1].

1 Introduction

Travel planning agents have become an essential role for assisting users in complex travel decision-making (Xie et al., 2024; Shao et al., 2026; Zhang et al., 2024; Ni et al., 2025; Deng et al., 2025). With the advancement of Large Language Models (LLMs) (Ma et al., 2025; He et al., 2025; Cheng et al., 2025a, 2026a; Gao et al., 2025), these agents are gaining the ability to interact with external environments to generate comprehensive travel plans (Ju et al., 2024; Luo et al., 2025; Zhang et al., 2025; Singh et al., 2025; Zhang et al., 2026), paving the way for more practical and intelligent travel solutions.

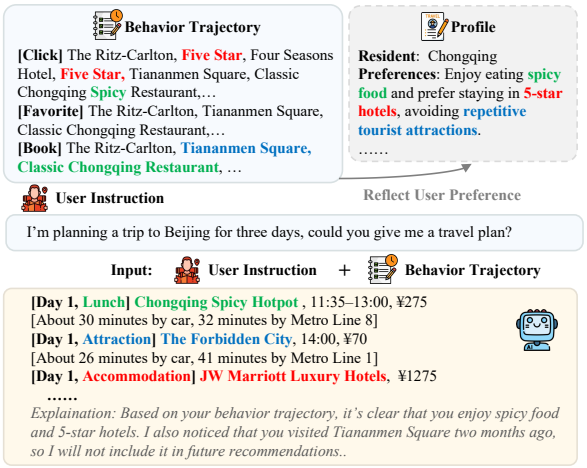

Figure 1: **An example of Behavior2Trip**, where the agent analyzes a user's behavioral trajectory to implicitly infer preferences, enabling the direct generation of personalized travel plans from implicit instructions.

Existing travel planning agents follow two main paradigms: **(1) Single-turn Explicit Instruction** (Xie et al., 2024; Shao et al., 2026; Ni et al., 2025; De La Rosa et al., 2024), which requires users to provide all preferences upfront in a single turn; and **(2) Multi-turn Clarification** (Zhang et al., 2024), which interactively elicits preferences through dialogue. However, both paradigms heavily rely on users' active input, overlooking behavioral signals from users' past interactions (e.g., clicks, favorites, and bookings) that naturally encode their preferences (Ye et al., 2011; Chen et al., 2026). This over-reliance on active input increases interaction burden and limits plan personalization.

[1]Code and data are available at https://github.com/BUAA-IRIP-LLM/Behavior2Trip.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 32110–32131
October 24-29, 2026 ©2026 Association for Computational Linguistics

Dataset	UBT	SII	OP	UPF	LIB	UP	# POIs	# POI Attr.	# Constraints	Instances
TravelPlanner (Xie et al., 2024)	✗	✗	✓	✗	✗	exp.	19.9k	6.6	15	20
ChinaTravel (Shao et al., 2026)	✗	✗	✓	✗	✗	exp.	12.2k	6.8	23	154
Ask–before–plan (Zhang et al., 2024)	✗	✓	✗	✗	✗	exp.	19.9k	6.6	12	2,000
ITINERA (Tang et al., 2024)	✗	✗	✓	✗	✗	exp.	7.6k	9.0	–	–
TRIP–PAL (De La Rosa et al., 2024)	✗	✗	✓	✗	✗	exp.	1.8k	–	2	100
Travel–Agent (Chen et al., 2024)	✗	✗	✗	✗	✗	exp.	–	–	20	20
TP–RAG (Ni et al., 2025)	✗	✗	✓	✗	✗	exp.	85.6k	–	–	2,348
Behavior2Trip (ours)	✓	✓	✓	✓	✓	**imp.**	**80.9k**	**17.7**	**32**	**11,400**

Table 1: **Comparison between the Behavior2Trip and other benchmarks. UBT**: User Behavior Trajectory, **SII**: Support Implicit Instruction, **OP**: One–shot Planning, **UPF**: User Profile, **LIB**: Low Interaction Burden, **UP**: User Preference. The ✓ indicates full support, the ✗ indicates no support, "–" denotes not mentioned in paper.

To bridge this gap, we introduce a new task, **Behavior-Aware Travel Planning**, which infers user preferences directly from past behaviors and generates personalized travel plans without requiring active user input. To support this task, we construct **Behavior2Trip**, a benchmark grounded in real user data from one of the largest Chinese online travel platforms, comprising 11,400 instances, each with an average of 39.8 past user behaviors spanning 14 attributes across 5 preference dimensions. The dataset is built upon a sandbox environment with 80.6k real-world POIs, where behavior trajectories are generated from diverse preference profiles and validated through a rigorous two-stage quality control process.

To address the challenge of implicitly inferring user preferences from behavior trajectories while satisfying long-horizon planning constraints, we propose **B2T-Agent**, a reinforcement learning-based agent that generates personalized travel plans from user behavior trajectories. Unlike prompt-driven methods (Yuan et al., 2025; Xie et al., 2024), B2T-Agent features a structured action space supporting both **external tool invocation** and **internal memory management**, optimized by a composite reward function that rewards plan quality and penalizes invalid actions.

We conduct comprehensive experiments on Behavior2Trip to evaluate both open-sourced and closed-sourced models. Results highlight the substantial challenge of this task: even GPT-4.1 achieves a full-constraint pass rate of only 0.5% on the hardest tasks, while B2T-Agent built upon Qwen3-8B significantly outperforms all baselines. Moreover, Qwen3-8B trained with B2T-Agent also outperforms GPT-4.1 on the TravelPlanner (Xie et al., 2024), demonstrating strong generalization. Overall, the contributions of this paper are as follows:

- We identify a novel task, **Behavior-Aware Travel Planning**, which generates personalized travel plans by inferring user preferences directly from past behaviors, without requiring explicit or iterative user input.
- To support this task, we introduce **Behavior2Trip**, a large-scale benchmark grounded in real user data, comprising 11,400 instances with rich behavioral signals spanning 14 attributes across 5 preference dimensions.
- To tackle the challenges of this task, we propose **B2T-Agent**, a reinforcement learning-based agent with structured external tool invocation and internal memory management. Experiments show that B2T-Agent built upon Qwen3-8B significantly outperforms GPT-4.1 and generalizes well to the TravelPlanner benchmark.

2 Related Work

2.1 Travel Planning Benchmarks

Travel planning is a representative task for evaluating agents' long-horizon planning capabilities, as it requires understanding user instructions, performing multi-turn tool use to gather information, and generating constraint-satisfying travel plans. Existing benchmarks fall into two paradigms: **(1) Single-turn Explicit Instruction** (Xie et al., 2024; Shao et al., 2026; De La Rosa et al., 2024; Singh et al., 2024), where the agent generates a complete travel plan from a single, explicit user request; and **(2) Multi-turn Clarification** (Zhang et al., 2024), where the agent interactively elicits user preferences through dialogue to refine the plan. However, as shown in Table 1, both paradigms require users to actively express their preferences and cannot address scenarios where users express their preferences implicitly. In contrast, **Behavior2Trip** evaluates agents' ability to infer user preferences

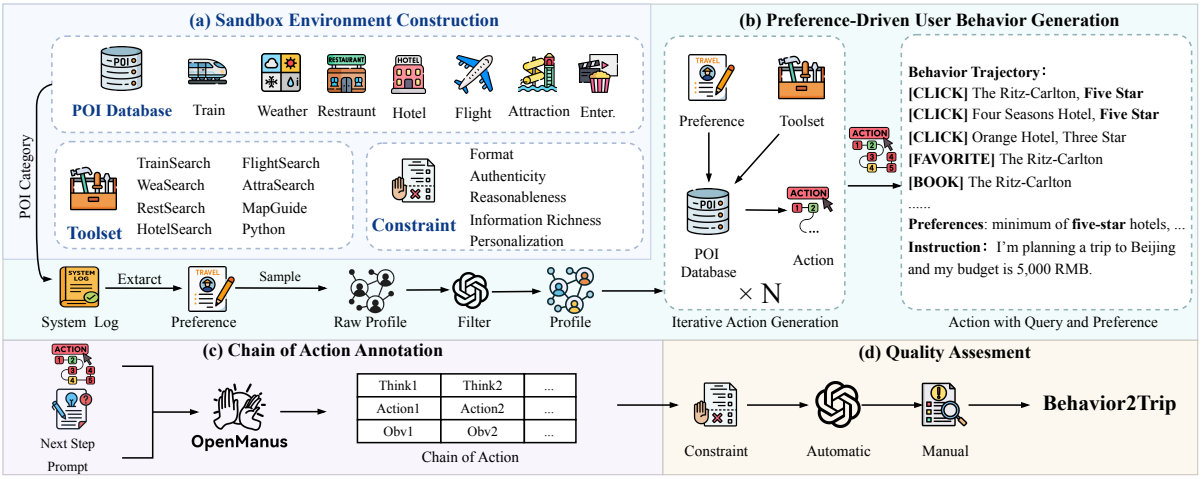

Figure 2: **The overall construction pipeline of Behavior2Trip**, comprising four phases: (a) **Sandbox Environment Construction** builds a realistic platform with real-world POIs, tools, and constraints; (b) **User Behavior Generation** produces user trajectories across three difficulty levels; (c) **Chain of Action Annotation** collects agent trajectories with chain-of-action reasoning paths; and (d) **Quality Assessment** ensures data quality through two-stage automated and human verification.

from behavior trajectories and generate preference-aligned travel plans, enabling more realistic evaluation of personalized planning in real-world scenarios.

2.2 Travel Planning Agents

To build travel planning agents, existing methods generally fall into two paradigms. **(1) End-to-End LLM Agent** (Xie et al., 2024; Zhang et al., 2024; Tang et al., 2024; Liu et al., 2026; Lan et al., 2026), which relies on manually designed prompts to guide information retrieval and plan generation. However, such handcrafted workflows lack flexibility and struggle to generalize across diverse user inputs. **(2) Hybrid LLM-Symbolic Solver** (Hao et al., 2025; Ju et al., 2024; De La Rosa et al., 2024; Shao et al., 2026), which improves logical consistency by incorporating symbolic solvers, but typically applies them only at the final planning stage, leaving the step-by-step information gathering process unguided. To address these limitations, we propose **B2T-Agent**, a reinforcement learning-based agent that eliminates handcrafted components and learns to interact with external tools and manage internal memory throughout the entire planning process.

3 Behavior2Trip

3.1 Problem Definition

We define the user's action trajectory as $\mathcal{A} = [a_1, a_2, \ldots, a_t]$, where each a_i denotes an action (e.g., click, favorite, book) at time step i, implicitly encoding the user's travel preferences $\mathcal{P}$. Given a natural-language travel request $\mathcal{I}$, the model queries the POI database $\mathcal{D}$ via toolset $\mathcal{T}$ to generate a personalized travel plan $\mathcal{R} = \text{Model}(\mathcal{A}, \mathcal{I}, \mathcal{T}, \mathcal{D})$, subject to two constraints: ① **Commonsense Constraint (CC)** ($\mathcal{R} \in \mathcal{C}_{CC}$): the plan must be logically feasible and factually grounded; ② **User Preference Constraint (UPC)** ($\mathcal{R} \in \mathcal{C}_{UPC}$): the plan must align with the user's inferred preferences.

3.2 Data Collection

As shown in Figure 2, this section describes the step-by-step process of constructing a diverse and realistic dataset, which includes four key stages: (a) Sandbox Environment Construction, (b) Preference-Driven User Behavior Generation, (c) Chain of Action Annotation, and (d) Quality Assessment.

3.2.1 Sandbox Environment Construction

To create a realistic and fully functional sandbox environment, we establish a *POI Database*, a *Toolset*, and *Constraints* sourced from one of the largest Chinese online travel platforms. All data is fully anonymized with no user privacy information involved.

POI Database We collect 80.6k real POIs from 24 popular Chinese travel cities, including *Attraction*, *Hotel*, *Restaurant*, and *Entertainment*. Each POI has an average of 17.7 fields, including: ①

Statistic	Train			Test		
	E	M	H	E	M	H
	Dataset					
Samples	3000	3000	3000	800	800	800
	Trajectory					
Actions	14.5	38.2	66.7	14.2	37.6	67.2
Preferences	1.0	2.4	4.0	1.0	2.3	4.0
POIs	10.3	27.2	47.8	10.1	26.9	48.0
	Instruction					
Trip Duration	4.9	4.9	5.0	4.9	4.9	5.0
Companions	2.2	2.1	2.3	2.2	2.2	2.2

Table 2: **Statistics of Behavior2Trip.** E, M, and H represent Easy, Medium, and Hard levels, respectively.

basic info (e.g., name, category), ② spatiotemporal data (e.g., location, hours, pricing), ③ user feedback (e.g., reviews, ratings). We also include 728.4k records on *Flight*, *Train*, and *Weather*, forming a comprehensive POI database $\mathcal{D}$. See Appendix A.1 for details.

Toolset We design a fine-grained toolset where each POI category has a dedicated search tool supporting optional filtering fields such as location, price range, rating, and opening hours. We also implement *MapGuide*, which helps interpret spatial data, and *Python* for general-purpose computing. In total, we develop 9 tools; details are in Appendix A.2.

Constraints To enable fine-grained evaluation of travel plans while preserving real-world fidelity, we design a constraint set grounded in authentic online travel service scenarios. For **CC**, we define four dimensions: *Format* ensures structural consistency, *Authenticity* verifies POI-related correctness, *Reasonableness* checks logical feasibility, and *Information Richness* reflects plan completeness. For **UPC**, we introduce the *Personalization* dimension to capture alignment with user preferences. Altogether, we define 32 constraints; details are in Appendix A.3.

3.2.2 User Behavior Generation

To construct user trajectories that are both preference-grounded and behaviorally coherent, we first derive a structured preference schema from real logs, instantiate user profiles with controlled difficulty, and then use LLM role-playing to generate long-term click–favorite–book sequences.

Preference Schema We derive a structured preference schema from anonymized user logs by extracting **14** high-frequency attributes across **5** dimensions. For each attribute, we calibrate its value range and POI-category mapping against real-log distributions, with details in Appendix A.4.

User Profile We instantiate user profiles $\mathcal{P}$ at three difficulty levels to control preference complexity and behavioral noise. *Easy* includes one preference category without noise; *Medium* combines multiple categories with one noise behavior; and *Hard* covers all categories with a higher noise ratio. For each level, we sample attribute combinations from the schema and use GPT-4.1 to remove inconsistent profiles, with the prompt shown in Figure 8.

Behavior Trajectory Motivated by prior studies showing that user check-in trajectories are shaped by sequential, spatio-temporal, and semantic/category patterns (Cheng et al., 2013; Tian et al., 2023; Wang et al., 2022), we model real user sessions as coherent travel-intent trajectories. To capture this structure, for each profile $\mathcal{P}$, we construct a candidate pool containing preference-matched POIs and a difficulty-controlled set of noise POIs. Gemini-3-Flash is then prompted to role-play the user and generate coherent **click** $\rightarrow$ **add_to_favorites** $\rightarrow$ **book** sequences (prompt shown in Figure 9). Noise POIs are restricted to click actions to simulate exploratory browsing, while favorite and book actions remain preference-aligned. Repeating this process across sessions yields long-term trajectories $\mathcal{A}$, which are paired with real travel instructions $\mathcal{I}$ from system logs to form $(\mathcal{A}, \mathcal{P}, \mathcal{I})$ triplets.

3.2.3 Chain of Action Annotation

After obtaining the $(\mathcal{A}, \mathcal{P}, \mathcal{I})$ triplets, we annotate each instance with a **Chain of Action (CoA)** label that records the full reasoning trajectory—including tool calls and intermediate decisions—to provide step-level supervision for model training. To generate these trajectories, we use OpenManus (Liang et al., 2025) as the rollout harness, feeding $\mathcal{A}$ and $\mathcal{I}$ as inputs. Since unconstrained rollouts tend to drift from the correct planning direction, after each `(think, action, observation)` cycle we manually inject a tailored **Next Step Prompt** (Figure 7) to steer the agent toward the next correct step, ensuring the resulting trajectory remains coherent and usable as a supervision signal.

3.2.4 Quality Assessment

To ensure dataset quality, we use a two-stage filtering process (Liu et al., 2020, 2021; Cheng et al.,

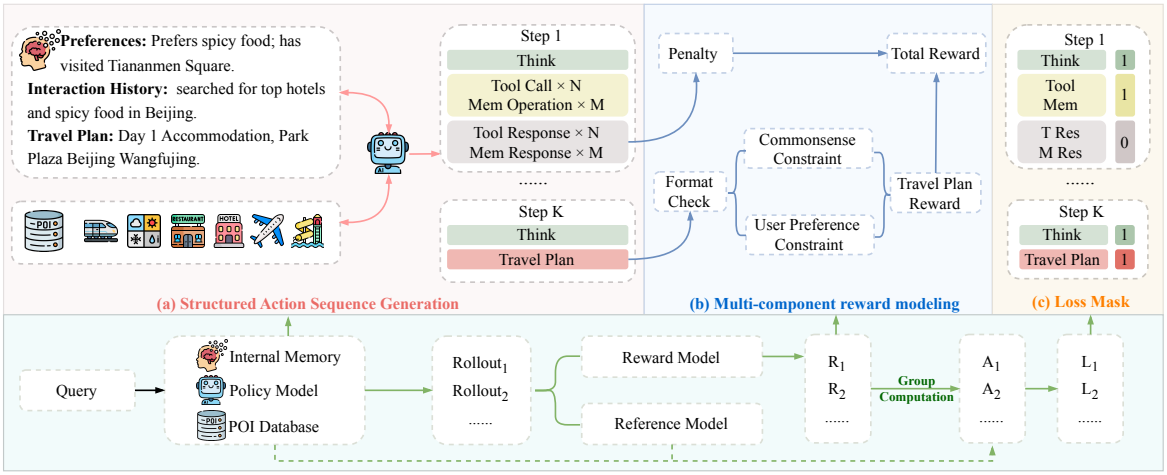

Figure 3: **Overview of the proposed B2T-Agent framework.** B2T-Agent incorporates **external tool invocation** and **internal memory management**, optimized via Group Relative Policy Optimization with a multi-component reward function that rewards plan quality and penalizes invalid actions.

2026b). In the first stage, we utilized Deepseek-R1 to automatically score each instance on a 0-3 scale with respect to three aspects: trajectory completeness, reflection of user preferences, and constraint satisfaction (Figure 10). Only instances that received a score of 3 were retained. Subsequently, three independent annotators reviewed a random sample of 1000 instances, reporting no observable errors and achieving 94.6% agreement.

3.3 Data Statistics

Table 2 summarizes dataset statistics across three difficulty levels. The full dataset comprises 11,400 instances (9,000 train / 2,400 test). With increasing difficulty, the number of actions, preference categories, and involved POIs all grow, making it progressively harder for models to infer user preferences from behavioral trajectories and generate constraint-satisfying plans. At each difficulty level, instructions span varied trip durations and group sizes, ensuring the benchmark captures a broad distribution of real-world travel planning scenarios.

4 B2T-Agent

In this section, we present B2T-Agent, an RL-based agent that supports external tool invocation and internal memory management for personalized travel planning. As shown in Figure 3, B2T-Agent comprises three key components: (a) Structured Action Sequence Generation (§4.1), (b) Multi-Component Reward Modeling (§4.2), and (c) Training with Loss Mask (§4.4).

4.1 Structured Action Sequence Generation

To support iterative POI retrieval from the database and coherent context management across long-horizon planning, we define the agent's rollout as a structured action sequence y:

$$y := (a_1, a_2, \ldots, a_k, R)$$

where each action a_i is one of the following types: External Tool Use $\mathcal{T}$, Internal Memory Management $\mathcal{M}$, or Chain-of-Thought Reasoning $\mathcal{C}$. The sequence concludes with a final Result $\mathcal{R}$.

External Tool Use The agent wraps each tool invocation in `<tool_call>` and receives the result in `<tool_response>`. Built on Qwen-Agent and following the Model Context Protocol[2], it can access any tool in our sandbox. To improve efficiency, multiple tools can be called within a single reasoning step.

Internal Memory Management To mitigate context window saturation from accumulating trajectories and tool responses, the agent maintains an internal memory module organized as key-value pairs, accessible via `<memory>`. The model can **read** entries by key and **write** new entries to store user preferences, tool outputs, and interaction history across planning steps.

Reasoning and Result To infer user preferences from behavior trajectories and guide subsequent

[2] https://www.anthropic.com/news/model-context-protocol

tool invocations, the model performs explicit reasoning within <think> to analyze retrieved information and intermediate results. Once all necessary interactions are complete, it generates the final travel plan within <answer>.

4.2 Multi-Component Reward Modeling

Given the structured action sequence defined above, we design a multi-component reward function to supervise both the process of tool invocation and the quality of the final travel plan, comprising an Incorrect Action Penalty and a Travel Plan Reward.

Incorrect Action Penalty To improve the reliability and robustness of the execution process, we introduce a penalty for invalid tool or memory invocations. An error is identified when <tool_response> or <memory_response> contains an explicit error message, upon which a penalty of $P_{\textbf{action}} = -1$ is applied to the entire sequence, and $P_{\textbf{action}} = 0$ otherwise.

Travel Plan Reward To evaluate the quality of generated travel plans against both commonsense and user preference constraints, we design a two-stage gating reward. A format score $R_{\textbf{format}} \in \{0, 1\}$ first verifies structural correctness; only if passed, the Commonsense Constraint score $R_{\textbf{CC}}$ and User Preference Constraint score $R_{\textbf{UPC}}$ are assessed, ensuring the model learns to produce well-formed outputs before optimizing for content quality:

$$R_{\textbf{plan}} = R_{\textbf{format}} \cdot (R_{\textbf{CC}} + R_{\textbf{UPC}})$$

The total reward combines both components:

$$R_{\textbf{total}} = P_{\textbf{action}} + R_{\textbf{plan}}$$

This design penalizes any action-level failure while rewarding high-quality, preference-aligned plans, jointly encouraging reliable execution and personalized planning.

4.3 Policy Optimization via GRPO

Inspired by recent advances in RL-based LLM training (Shan et al., 2026; Pan et al., 2025; Team et al., 2025; Wang et al., 2025, 2026; Liu et al.), we adopt GRPO (Shao et al., 2024) to optimize the policy model by maximizing the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{[\{y_i\}_{i=1}^{G} \sim \pi_\theta(Y|q)]} \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \Big\{ \min \left[r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right] \Big\}$$

$$\text{with} \quad r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|q, y_{i,<t})}{\pi_{\theta_{old}}(y_{i,t}|q, y_{i,<t})},$$

where π_θ and π_{ref} stand for the policy model and reference model respectively. $\mathbb{D}_{KL}$ is the KL-divergence, scaled by a coefficient β. q denotes a query, and y represents a sequence of structured actions.

4.4 Training with Loss Mask

During training, the rollout contains externally sourced tokens within <tool_response> and <memory_response> that are not generated by the model. Including them in the loss computation introduces noise, so we follow Jin et al. and apply a loss mask to exclude these segments from gradient updates, ensuring the model is optimized only on its own generated tokens.

5 Experiments

5.1 Experimental Setup

Baselines and Models Following prior works (Xie et al., 2024; Cheng et al., 2025b), we adopt two baselines: **ReAct** (Yao et al., 2023) and **SFT**, alongside our proposed **B2T-Agent**, all with interactive reasoning where the model interacts with the environment to gather information. To ensure fair comparison, all methods are given the full user behavior trajectory. We evaluate open-source models including Qwen3-8/14/32B (Yang et al., 2025) and Deepseek-V3 (Liu et al., 2024), as well as frontier proprietary models including GPT-4o and GPT-4.1.

Metrics Following Xie et al. (2024), we report the Delivery Rate (**DR**), which measures the proportion of plans that successfully produce a structurally valid travel itinerary. For constraint adherence, we compute the Micro Pass Rate (**Micro PR**) and Macro Pass Rate (**Macro PR**) separately for Commonsense Constraints (**CC**) and User Preference Constraints (**UPC**), defined as:

$$\text{Micro PR} = \frac{\sum_{p \in P} \sum_{c \in C_p} \mathbb{1}_{\text{passed}(c,p)}}{\sum_{p \in P} |C_p|}$$

Model	Method	Easy					Medium					Hard					Avg.
		DR	CC	UPC	Final	LLM	DR	CC	UPC	Final	LLM	DR	CC	UPC	Final	LLM	
Qwen3-8B	ReAct	40.2	34.2/0.0	10.1/10.1	0.0	27.5	34.1	31.8/0.0	12.3/0.0	0.0	23.5	11.2	24.1/0.0	15.9/0.0	0.0	12.8	0.0
Qwen3-14B		44.3	32.9/0.3	14.2/14.2	0.3	32.6	32.8	32.9/0.4	19.7/1.8	1.8	28.7	21.5	31.9/1.2	21.1/0.0	0.0	23.2	0.7
Qwen3-32B		53.1	32.4/0.3	14.3/14.3	0.3	34.3	41.5	32.5/0.8	19.8/0.8	0.8	29.9	31.4	32.4/1.3	22.7/0.0	0.0	33.3	0.4
GPT-4o		90.4	36.5/1.3	16.8/16.8	1.3	36.6	97.7	36.4/4.2	19.4/3.1	3.1	31.9	96.2	33.1/3.4	22.2/0.3	0.3	26.9	1.5
GPT-4.1		95.8	33.6/2.2	17.7/17.7	2.2	36.4	95.0	33.4/3.0	28.4/7.0	7.0	31.2	**97.8**	33.5/3.1	22.2/0.5	0.5	28.2	3.2
DeepSeek-V3		94.3	34.7/3.3	20.1/20.1	3.1	41.6	94.9	33.9/3.7	28.4/4.4	4.4	38.8	97.0	34.1/4.1	31.8/0.8	0.5	34.1	2.7
Qwen3-8B	SFT	77.2	54.0/5.0	17.7/17.7	5.0	73.7	76.5	54.4/4.6	26.5/3.7	3.7	62.3	57.4	54.5/4.1	32.9/0.3	0.3	35.8	3.0
Qwen3-14B		87.3	58.7/5.2	17.8/17.8	5.2	73.1	88.6	59.0/5.5	29.0/4.1	4.1	66.9	61.5	55.2/5.0	33.2/0.3	0.3	36.1	3.2
Qwen3-8B	B2T-Agent	**100.0**	56.8/5.2	25.0/25.0	4.7	77.2	96.3	57.3/5.2	37.2/10.3	5.2	70.2	93.3	53.3/5.1	36.9/4.3	4.3	68.5	4.7
Qwen3-14B		98.9	**59.8/7.3**	**27.9/27.9**	**7.3**	**81.1**	**99.0**	**59.3/6.2**	**40.3/13.5**	**6.2**	**75.2**	97.4	**59.4/6.1**	**41.2/5.0**	**5.0**	**74.3**	**6.2**

Table 3: **Performance on Different Methods and Difficulties.** CC and UPC show Micro/Macro Pass Rate. Results are reported in percentage (%). The best and second-best results are marked in bold and underlined.

$$\text{Macro PR} = \frac{\sum_{p \in P} \mathbb{1}_{\text{passed}(C_p, p)}}{|P|}$$

For overall performance, we report the Final Pass Rate (**Final**), the percentage of plans satisfying all constraints, and the LLM Pass Rate (**LLM**) for flexible LLM-based evaluation.

Implementation Details We adapt the ReAct baseline from TravelPlanner (Xie et al., 2024) to our expanded toolset and sandbox constraints, and conduct SFT using LLaMAFactory (Zheng et al., 2024). B2T-Agent is built on RLFactory (Chai et al., 2025) with the prompt template shown in Figure 11.

5.2 Main Results

Table 3 reports the main results and we highlight three key findings.

Existing models struggle with Action-Aware Travel Planning. Even the strongest proprietary model, GPT-4.1, achieves only 0.5 Final PR on Hard tasks, underscoring the severity of the challenge. This stems from a fundamental shift in what drives success: while Easy tasks can largely be solved by satisfying CC, Medium and Hard tasks increasingly require alignment with UPC, a more personalized and dynamic requirement that current models consistently fail to capture. While LLM-based evaluation assigns relatively higher scores than rule-based Final due to its tolerance for output variations, both metrics confirm the same trend: UPC satisfaction remains the key bottleneck as task difficulty scales up.

B2T-Agent achieves the best performance on Action-Aware Travel Planning. B2T-Agent consistently achieves the best performance across all difficulty levels, with B2T-Agent-14B leading all baselines on both Final PR and LLM PR across Easy, Medium, and Hard splits. Furthermore, this advantage becomes especially pronounced on harder tasks: on the Hard split, B2T-Agent-14B achieves 5.0 Final PR compared to 0.5 for the best ReAct baseline (GPT-4.1), and 74.3 LLM PR compared to 28.2. Notably, even B2T-Agent with Qwen3-8B surpasses GPT-4.1 in average Final PR (4.7 vs. 3.2), highlighting the efficiency of our approach.

B2T-Agent Achieves Superior Performance on User Preference Constraints. Since CC constraints are shared across all queries, SFT can learn fixed patterns to improve CC satisfaction; however, UPC constraints vary per user and require dynamic adaptation, which SFT fails to provide. In contrast, B2T-Agent leverages reward signals to drive the model toward personalized planning through interaction, rather than relying on memorized response patterns. This translates to substantial UPC gains: on the Hard split, B2T-Agent-14B achieves 5.0 UPC Macro PR, compared to 0.3 for SFT-14B and 0.8 for the best ReAct baseline (DeepSeek-V3).

6 Analysis

In this section, we present a comprehensive analysis aimed at addressing the following research questions. **RQ1:** *Does incorporating behavior trajectory really help with personalized travel planning?* (§6.1) **RQ2:** *How does increasing trajectory complexity affect method performance?* (§6.2) **RQ3:** *How does each component affect the performance of B2T-Agent?* (§6.3) **RQ4:** *Can B2T-Agent generalize to other travel planning benchmarks?* (§6.4)

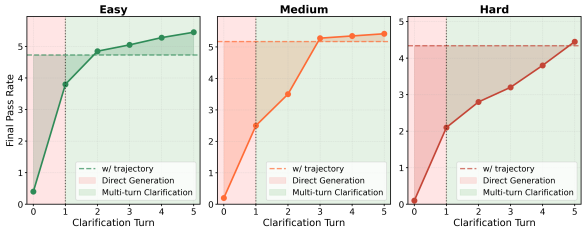

Figure 4: **Impact of Trajectory and Clarification Turns on Final PR.** Incorporating user trajectory significantly improves initial plan quality and reduces the need for clarification turns.

6.1 RQ1: Impact of Behavior Trajectory

To investigate whether user behavior trajectory benefits personalized travel planning, we evaluate Qwen3-8B-B2T-Agent using only implicit user instructions, without trajectory. Instead, GPT-4.1 is used to simulate user feedback, allowing the model to iteratively refine plans through clarification turns. As shown in Figure 4, with zero clarification turns, Final PR is nearly zero, indicating failure to align plans with user preferences. To match the performance of trajectory-augmented planning, Easy, Medium, and Hard tasks require approximately 2, 3, and 5 clarification turns, respectively. These results confirm that user trajectory substantially improves initial plan quality and reduces reliance on multi-turn clarification.

6.2 RQ2: Impact of Trajectory Complexity

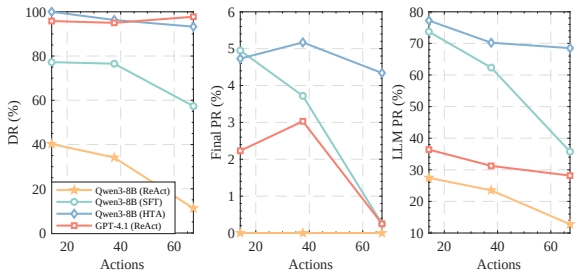

Figure 5: **Relationship between the number of actions in trajectories and the performance** of DR, Final PR, and LLM PR.

We compare Qwen3-8B and GPT-4.1 across DR, Final PR, and LLM PR as trajectory complexity increases (measured by the number of actions). As shown in Figure 5, Qwen3-8B under SFT and ReAct shows sharp drops across all metrics, as longer trajectories introduce more user preferences and edge cases beyond their capacity. In contrast, GPT-4.1 maintains strong DR but suffers notable declines in Final PR and LLM PR. Qwen3-8B-B2T-Agent performs best overall, sustaining high DR and stable PR scores; reward-driven interactive training enables it to handle edge cases and capture user preference–POI relationships that SFT and prompting cannot cover.

6.3 RQ3: Ablation Study

Method	DR	Final PR	LLM PR
B2T-Agent	96.52	**4.75**	**71.98**
- *w/o. M*	**97.46** (↑0.97%)	4.21 (↓11.37%)	64.24 (↓10.76%)
- *w/o. P*	86.70 (↓10.17%)	4.32 (↓9.05%)	65.22 (↓9.39%)
- *w/o. L*	78.18 (↓18.99%)	3.79 (↓20.21%)	57.97 (↓19.47%)

Table 4: **Ablation study on Qwen3-8B.**

We ablate each key component of B2T-Agent on Qwen3-8B to assess their individual contributions. As shown in Table 4, removing the memory module (*w/o. Memory*) slightly improves DR (+0.97%), as the model skips memory operations and favors faster responses. However, without memory, the model loses access to user preferences, causing Final PR and LLM PR to drop by 11.4% and 10.8%, respectively. Without the penalty term (*w/o. Penalty*), the model fails to correct inappropriate tool or memory actions, leading to repetitive loops and a 10.2% drop in DR. Finally, removing loss masking (*w/o. Loss Mask*) introduces noisy gradients from non-answer tokens, causing the largest overall decline, with DR, Final PR, and LLM PR dropping by 19.0%, 20.2%, and 19.5%, respectively.

6.4 RQ4: Cross-Benchmark Evaluation

Method	DR	CMi	CMa	HMi	HMa	FPR
Mistral-7B†	7.0	4.8	0.0	0.0	0.0	0.0
GPT-4-Turbo†	93.1	63.3	2.9	10.5	5.5	0.6
GPT-4.1	99.4	75.1	5.7	10.7	7.8	1.5
Qwen3-8B-B2T-Agent	**100**	**76.8**	**25.0**	**26.5**	**10.2**	**9.0**

Table 5: **Performance on the TravelPlanner** (†results from the original paper).

To validate the generality of B2T-Agent for travel planning, we further conduct a cross-benchmark evaluation on TravelPlanner (Xie et al., 2024). As shown in Table 5, Qwen3-8B-B2T-Agent consistently outperforms all baselines, with especially large gains on constraint satisfaction (CMa: 25.0 vs. 5.7) and overall plan quality (FPR: 9.0 vs. 1.5). These results confirm that B2T-Agent generalizes effectively beyond our benchmark to standard travel planning settings.

7 Conclusion

In this paper, we identify a new task, Behavior-Aware Travel Planning, which infers user preferences from behavioral trajectories to generate personalized travel plans without requiring explicit input. To support this task, we present Behavior2Trip, a benchmark combining user behavior data, and B2T-Agent, a reinforcement learning framework that enables models to interact autonomously using external tools and internal memory. Extensive experiments demonstrate that B2T-Agent effectively captures user intent, significantly reducing the need for explicit user interaction and enabling more seamless travel planning experiences.

Limitations

While B2T-Agent achieves high-quality personalized planning by combining reward-driven multi-turn tool interaction with reinforcement learning, it incurs notable computational overhead from two main sources. First, during training, GRPO requires sampling multiple rollouts per query, which substantially increases computational cost. Second, at inference time, the multi-turn tool invocations lead to high first-token latency. Reducing this overhead without compromising planning quality remains an important direction for future work.

Ethics Statement

POI and user-preference data can contain substantial sensitive information (e.g., consumers' personally identifiable information), making it essential to protect user privacy from both the data and model perspectives. Specifically, as described in Section 3.2.1, we first remove all explicit personally identifiable information from the POI database by using a locally deployed GPT-oss-120B (Agarwal et al., 2025) to redact user reviews, business phone numbers, and POI descriptions. We then conduct random manual audits to verify that no privacy-sensitive content remains. In addition, during preference extraction, we model only high-level attributes rather than individual users, thereby further reducing privacy risk and mitigating potential bias when learning the mapping between preferences and POI categories.

Acknowledgments

Thanks for the insightful comments and feedback from the reviewers. This work was supported by the National Natural Science Foundation of China (No. 62406015).

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, et al. 2025. Rlfactory: A plug-and-play reinforcement learning post-training framework for llm multi-turn tool-use. *arXiv preprint arXiv:2509.06980*.

Aili Chen, Xuyang Ge, Ziquan Fu, Yanghua Xiao, and Jiangjie Chen. 2024. Travelagent: An ai assistant for personalized travel planning. *Preprint*, arXiv:2409.08069.

Xiaoqing Chen, Zhitao Li, Weike Pan, Qiang Yang, and Zhong Ming. 2026. A survey on multi-behavior sequential recommendation. *Science China Information Sciences*, 69(3):131101.

Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, volume 13, pages 2605–2611.

Zhangtao Cheng, Yuhao Ma, Jian Lang, Kunpeng Zhang, Ting Zhong, Yong Wang, and Fan Zhou. 2025a. Generative thinking, corrective action: User-friendly composed image retrieval via automatic multi-agent collaboration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 334–344.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. 2026a. Mem2evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20784–20831.

Zihao Cheng, Hongru Wang, Zeming Liu, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. 2025b. Toolspectrum: Towards personalized tool utilization for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20679–20699.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. 2026b. Terminal-world: Scaling terminal-agent environments via agent skills. *Preprint*, arXiv:2605.20876.

Tomas De La Rosa, Sriram Gopalakrishnan, Alberto Pozanco, Zhen Zeng, and Daniel Borrajo. 2024. Trippal: Travel planning with guarantees by combining large language models and automated planners. *arXiv preprint arXiv:2406.10196*.

Bin Deng, Yizhe Feng, Zeming Liu, Qing Wei, Xiangrong Zhu, Shuai Chen, Yuanfang Guo, and Yunhong Wang. 2025. Retail: Towards real-world travel planning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14881–14913.

Fan Gao, Hongqiang Li, Zhilong Chen, Yunai Yi, Shihao Nie, Zihao Cheng, Zeming Liu, Yuanfang Guo, Shumin Liu, Qizhen Qin, et al. 2025. A chemical autonomous robotic platform for end-to-end synthesis of nanoparticles. *Nature Communications*, 16(1):7558.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2025. Large language models can solve real-world planning rigorously with formal verification tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3434–3483.

Yingzhi He, Xiaohao Liu, An Zhang, Yunshan Ma, and Tat-Seng Chua. 2025. Llm2rec: Large language models are powerful embedding models for sequential recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 896–907.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*.

Da Ju, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T Kao, Maryam Fazel-Zarandi, et al. 2024. To the globe (ttg): Towards language-driven guaranteed travel planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 240–249.

Tianwei Lan, Jiaqi Wu, Zeming Liu, Zhaoxin Fan, Haifeng Wang, and Yuhang Guo. 2026. Peap: Proactive embodied action sequence planning with joint understanding of vision and audio perception. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23118–23138.

Xinbing Liang, Jinyu Xiang, Zhaoyang Yu, Jiayi Zhang, Sirui Hong, Sheng Fan, Xiao Tang, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025. Openmanus: An open-source framework for building general ai agents.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Jingjing Liu, Ziye Huang, Zihao Cheng, Zeming Liu, Jiahong Wu, Yuhang Guo, Kehai Chen, Yunhong Wang, and Haifeng Wang. Docos: Towards proactive document-guided actions in gui agents. In *Forty-third International Conference on Machine Learning*.

Jingjing Liu, Silin Li, Zeming Liu, Zihao Cheng, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. 2026. Towards multi-language repository-level code generation: From-scratch to guided tasks. *Neurocomputing*, page 133204.

Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, and Wanxiang Che. 2021. Durecdial 2.0: A bilingual parallel corpus for conversational recommendation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4335–4347.

Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Towards conversational recommendation over multi-type dialogs. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 1036–1049.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Zhiyuan Ma, Jiayu Liu, Xianzhen Luo, Zhenya Huang, Qingfu Zhu, and Wanxiang Che. 2025. Advancing tool-augmented large language models via meta-verification and reflection learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 2078–2089.

Hang Ni, Fan Liu, Xinyu Ma, Lixin Su, Shuaiqiang Wang, Dawei Yin, Hui Xiong, and Hao Liu. 2025. Tp-rag: Benchmarking retrieval-augmented large language model agents for spatiotemporal-aware travel planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12403–12429.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. Tinyzero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Yingyu Shan, Yuhang Guo, Zihao Cheng, Zeming Liu, Xiangrong Zhu, Xinyi Wang, Jiashu Yao, Wei Lin, Hongru Wang, and Heyan Huang. 2026. Learning from own solutions: Self-conditioned credit assignment for reinforcement learning with verifiable rewards. *Preprint*, arXiv:2606.18810.

Jie-Jing Shao, Bo-Wen Zhang, Xiao-Wen Yang, Baizhi Chen, Siyu Han, Jinghao Pang, Wen-Da Wei, Guohao

Cai, Zhenhua Dong, Lan-Zhe Guo, et al. 2026. Chinatravel: an open-ended travel planning benchmark with compositional constraint validation for language agents. In *International Conference on Learning Representations*, volume 2026, pages 96008–96060.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 2024. Personal large language model agents: A case study on tailored travel planning. In *Proceedings of the 2024 conference on empirical methods in natural language processing: industry track*, pages 486–514.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *Preprint*, arXiv:2505.01441.

Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Zhaofeng Wu, Dingyi Zhuang, Jushi Kai, Kebing Hou, Xiaotong Guo, Jinhua Zhao, et al. 2024. Itinera: Integrating spatial optimization with large language models for open-domain urban itinerary planning. In *Proceedings of the 2024 conference on empirical methods in natural language processing: Industry track*, pages 1413–1432.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Jing Tian, Zilin Zhao, and Zhiming Ding. 2023. Next point-of-interest recommendation based on joint mining of spatial–temporal and semantic sequential patterns. *ISPRS International Journal of Geo-Information*, 12(7):297.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*.

Xican Wang, Yanheng Liu, Xu Zhou, Xueying Wang, and Zhaoqi Leng. 2022. A point-of-interest recommendation method exploiting sequential, category and geographical influence. *ISPRS International Journal of Geo-Information*, 11(2):80.

Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru Wang, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, et al. 2026. Perception-aware policy optimization for multimodal reasoning. In *International Conference on Learning Representations*, volume 2026, pages 80502–80530.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, pages 54590–54613.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 325–334.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2025. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6192–6217.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2026. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. In *International Conference on Learning Representations*, volume 2026, pages 91437–91453.

Xuan Zhang, Yang Deng, Zifeng Ren, See Kiong Ng, and Tat-Seng Chua. 2024. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the association for computational linguistics: EMNLP 2024*, pages 10836–10863.

Yuxiang Zhang, Yuqi Yang, Jiangming Shu, Xinyan Wen, and Jitao Sang. 2025. Agent models: Internalizing chain-of-action generation into reasoning models. *arXiv preprint arXiv:2503.06580*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pages 400–410.

A Behavior2Trip

A.1 POI Database Details

Table 6 presents all the real Points of Interest (POIs) information used in this paper. It includes the POI types, the number of POIs of each kind, and the names of fields associated with each type. This comprehensive dataset provides a realistic and diverse sandbox environment for our research.

A.2 Toolset Details

Table 7 presents all the tools, including the tool names, descriptions, and their respective parameters. A total of nine tools are provided, each supporting fine-grained combinations based on user preference information, enabling precise retrieval of the most relevant data from the large POI database.

A.3 Constraint Details

Table 8 lists the constraints used to evaluate the quality of travel plans. A total of 32 constraints are categorized into five groups: Format (1), Authenticity (4), Reasonableness (15), Personalization (10), and Information Richness (2). Each constraint is derived from real-world business scenarios, providing a realistic and effective basis for assessing the quality of a travel plan.

A.4 Preference Details

Table 9 presents the fields and value ranges related to travel preferences defined in user profiles, including Historical Travel Experiences, Hotel Preferences, Attraction Preferences, Dining Requirements, and Transportation Preferences. Each field is closely aligned with POIs in the database, ensuring that user instructions with diverse preferences can be effectively addressed.

B Additional Experimentss

B.1 RQ5: Case Study

As illustrated in Figure 6, we compare the performance of B2T-Agent and TravelPlanner ReAct when processing the same input. The ReAct model performs tool invocation in a strictly sequential manner, relying entirely on prompts to execute actions step by step. As the number of turns increases, the dialogue context grows rapidly, making it difficult for the model to retain previous interaction history and user preferences. This often leads to repeated tool usage and a lack of filtering based on user preferences. Such inefficiency not only increases the number of interaction turns but may also prevent the generation of a final travel plan due to the maximum turn limit.

In contrast, B2T-Agent supports parallel tool invocation and incorporates a memory module to read and write critical user information. During the planning process, the model proactively accesses stored user preferences and interaction history, thereby improving tool invocation efficiency and generating travel plans that better align with user needs.

B.2 RQ6: Error Analysis

We perform a detailed error analysis on the two best-performing models from our main experiment and our proposed B2T-Agent. From each model's output, we randomly sampled 30 generated trajectories and manually examined them (total of 180 samples), identifying and categorizing five classes of recurring errors.

B.2.1 *Constraint Violations*

Agents often generate travel plans that ignore real-world constraints, such as geographic feasibility, time budgets, and user preferences. While the outputs usually satisfy format and completeness requirements, they fail to consider whether the recommended POIs are realistically visitable or aligned with user needs. As a result, most plans are not directly usable.

B.2.2 *Erroneous Tool or Memory Loops*

When interacting step-by-step with external tools or internal memory, the models sometimes generate invalid tool call formats or incorrect parameters. Once an error occurs, the model does not revise its behavior based on feedback, leading to repeated failures and ultimately crashing the generation process.

B.2.3 *Hallucinated POIs*

Agents occasionally include POIs that do not exist in the database. This type of hallucination indicates a poor understanding of the model's knowledge boundaries. Instead of retrieving POIs from verified data, the model relies on memorized or incorrect internal representations.

B.2.4 *Contextual Reasoning Limits*

Each generation involves extensive interaction with memory and environment, leading to long contextual sequences, often exceeding 20k tokens. The

Behavior Trajectory

[{“action_type”:“click”,“timestamp”:“2024-06-05T19:51:49”,“poi_details”:{“poi_type”:“Food & Beverage”,“name”:“Junkang Yayuan Rooftop Garden Restaurant”,“subcategory”:“Hunan Cuisine”}},{“action_type”:“click”,“timestamp”:“2024-06-05T19:51:50”,“poi_details”:{“poi_type”:“Food & Beverage”,“name”:“Old Hunan Cuisine (Denggao Road Branch)”,“subcategory”:“Hunan Cuisine”}},{“action_type”:“book”,“timestamp”:“2024-06-05T19:52:02”,“poi_details”:{“poi_type”:“Food & Beverage”,“name”:“Authentic Huofu Xiangtan Cuisine (Hexi Wangfujing Branch)”,“subcategory”:“Hunan Cuisine”}},{“action_type”:“click”,“timestamp”:“2024-04-23T21:53:52”,“poi_details”:{“poi_type”:“Hotel”,“name”:“Nanjing Fengda International Hotel”,“primary_category”:“Hotel”,“secondary_category”:“Five-Star/Luxury”,“tertiary_category”:“Luxury”}},{“action_type”:“book”,“timestamp”:“2024-04-23T21:54:04”,“poi_details”:{“poi_type”:“Hotel”,“name”:“Beijing Baihewang Chaolin Songyuan Hotel”,“primary_category”:“Hotel”,“secondary_category”:“Five-Star/Luxury”,“tertiary_category”:“Luxury”}}, ……]

User Instruction

I’m planning a trip to Beijing for three days, could you give me a travel plan?

<|im_start|> assistant
<think> Okay, the user has now requested a three-day trip to Beijing. First, I need to analyze the user’s preferences based on their behavior trajectory. It’s clear that the user prefers spicy food, and the hotel history shows only 5-star or above hotels. I will first record the user’s preferences and then search for dining, lodging, and attractions in Beijing. </think>
<memory>write(key=‘preference’, value={‘restaurant’: ‘spicy’, ‘hotel_requirement’: ‘5-star’})</memory>
<tool_call>{‘name’: ‘accommodationsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:5, ‘preference’:{‘Star Rating Requirement’:’Five-Star/Luxury’}]}</tool_call>
<tool_call>>{‘name’: ‘restaurantsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:10, ‘preference’:{‘Preferred Cuisine’:’Hunan Cuisine’}]}</tool_call>
<tool_call>>{‘name’: ‘attractionsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:10]}</tool_call> <|im_emd|>

<|im_start|>user
< memory _response>Data saved for key ‘preference’.</ memory _response>
<tool_response> … </tool_response>
<tool_response> … </tool_response>
<tool_response> … </tool_response><|im_end|>

…….

<|im_start|> assistant
<think>Okay, the three tools were successfully called in the last step. We’ll select The Forbidden City, JW Marriott Luxury Hotels, and Chongqing Spicy Hotpot as the POIs for Day 1. At the same time, I will use MapGuide to query the route.</think>
<memory>write(key=‘day1’, value={‘accommodation’: ‘JW Marriott Luxury Hotels’, ‘attraction’:’The Forbidden City’, ‘restaurant’: ‘Chongqing Spicy Hotpot’})</memory>
<tool_call>>{‘name’: ‘mapguide’, ‘arguments’: [‘origin’: ‘The Forbidden City’, ‘destination’: ‘JW Marriott Luxury Hotels’]}</tool_call><|im_emd|>

<|im_start|> user
< memory _response>Data saved for key ‘day1’.</ memory _response>
<tool_response> … </tool_response><|im_end|>

…….

<|im_start|> assistant
<think>I need to check the POIs I’ve already collected and confirm the user’s preferences again…</think>
<memory>read(key=‘all’)</memory>

<|im_start|> user
< memory _response>{‘restaurant’: ‘spicy’, ‘hotel_requirement’: ‘5-star’, ‘day1’: {‘accommodation’: ‘JW Marriott Luxury Hotels’, ‘attraction’:’The Forbidden City’, ‘restaurant’: ‘Chongqing Spicy Hotpot’ },…….}</ memory _response> <|im_end|>

…….

<|im_start|> assistant
<think>I now have enough information. From the memory response, I can see the user likes spicy food and prefers 5-star hotels. Let me generate the full plan for the user in one go.</think>

<result>{“Day 1”:{“Lunch”:{“name”:“Chongqing Spicy Hotpot”,“time”:“11:35–13:00”,“price”:275,“transport”:{“car”:“~30 minutes”,“metro”:“Line 8 ~32 minutes”}},“Attraction”:{“name”:“The Forbidden City”,“time”:“14:00”,“price”:70,“transport”:{“car”:“~26 minutes”,“metro”:“Line 1 ~41 minutes”}},“Accommodation”:{“name”:“JW Marriott Luxury Hotels”,“price”:1275}}, ……}</result> <|im_end|>

B2T-Agent

Thought1: The user wants to plan a three-day trip to Beijing. According to the user's request, I will first analyze the user's preferences based on the behavior trajectory. The user likes spicy food and wants to stay in 5-star hotels. Let me first collect POI information on 5-star hotels in Beijing.
Action1:{‘name’: ‘accommodationsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:5, ‘preference’:{‘Star Rating Requirement’:’Five-Star/Luxury’}]}
Observation1: …

Thought2: Then, I will continue collecting restaurant information and pay attention to the user’s food preferences — the user prefers Hunan cuisine.
Action2: {‘name’: ‘restaurantsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:10, ‘preference’:{‘Preferred Cuisine’:’Hunan Cuisine’}]}
Observation2: …

Thought3: I have already collected restaurant and accommodation information. Now I need to plan the first day’s attractions.
Action3: {‘name’: ‘attractionsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:1]}
Observation3: …

Thought4: I already have enough information to create the plan for the first day. I will choose Chongqing Spicy Hotpot for lunch, and arrange The Forbidden City and JW Marriott Luxury Hotels as the attraction and accommodation. According to the user’s request, let’s continue planning the city transportation route.
Action4: {‘name’: ‘mapguide’, ‘arguments’: [‘origin’: ‘the forbidden city’, ‘destination’: ‘JW Marriott Luxury Hotels’]}
Observation4: …

……

Thought20: Now I will collect hotel information for the first day.
Action20: {‘name’: ‘accommodationsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:5]}
Observation20: …

Forget interaction history and user preferences.

Thought21: I will continue collecting restaurant information.
Action21: {‘name’: ‘restaurantsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:10, ‘preference’:{‘Preferred Cuisine’:’Hunan Cuisine’}]}
Observation21: …

Thought21: Continue searching for attractions in Beijing.
Action21: {‘name’: ‘attractionsearch’, ‘arguments’: [‘city’: ‘beijing’, ‘n’:1]}
Observation21: …

…

Exceeded the maximum number of 25 turns, unable to provide the final plan.

Serial execution is inefficient and repeatedly collects already available information, resulting in exceeding the maximum number of turns.

Travelplanner ReAct

Figure 6: Case study comparing the reasoning trajectories of TravelPlanner ReAct and B2T-Agent. While the ReAct baseline suffers from context loss and redundant tool usage due to strictly sequential execution, B2T-Agent utilizes parallel tool invocation and a memory module to retrieve user preferences, ensuring efficient and accurate plan generation.

POI Type	Count	Features
Restaurant	29,604	*Name, Details, Address, Phone, Province, City, Location, Longitude, Latitude, Tertiary Category, Secondary Category, Primary Category, Business Hours, Rating Rank, Parking Information, Average Price, Average Rating, User Reviews*
Flights	710,633	*Flight Number, Airline, Airline Code, Departure City, Arrival City, Departure City Code, Arrival City Code, Departure Airport, Arrival Airport, Departure Airport Code, Arrival Airport Code, Departure Date, Arrival Date, Departure Time, Arrival Time, Flight Duration, Departure Terminal, Arrival Terminal, Aircraft Model, Cabin Type, Cabin Code, Adult Fare, Child Fare, Adult Total Price, Child Total Price, Adult Fuel Surcharge, Adult Airport Construction Fee, Discount, On-time Rate, Meal Availability, Codeshare Flight*
Hotels	22,473	*Name, Details, Cover Image, Address, Phone, Province, City, District, Longitude, Latitude, Tertiary Category, Secondary Category, Primary Category, Business Hours, Rating Rank, Parking Information, WiFi Availability, Average Price, Average Rating, User Reviews, Room Information*
Entertainment	20,250	*Name, Details, Cover Image, Address, Phone, Province, City, District, Longitude, Latitude, Tertiary Category, Secondary Category, Primary Category, Business Hours, Rating Rank, Average Price, Average Rating*
Trains	10,371	*Departure City, Arrival City, Departure Station, Arrival Station, Train Number, Train Type, Travel Time, Travel Minutes, Departure Time, Arrival Time, Fare*
Attractions	8,229	*Name, Details, Cover Image, Address, Phone, Province, City, Location, Longitude, Latitude, Category Name, Attraction Type, Custom Type, Opening Hours, Best Visiting Season, Recommended Visiting Duration, Average Price, User Reviews, Ticket Information*
Weather	7,440	*City, Date, Max Temperature, Min Temperature, Weather, Wind Direction, Air Quality*

Table 6: Statistics of the collected **real-world** POI database. The dataset comprises **7** distinct categories totaling **809k** records, with an average of **17.7** attribute features per category.

models struggle to extract relevant signals from such complex contexts, which negatively impacts their ability to generate coherent and relevant plans.

B.2.5 *Formatting Errors*

Despite clear instruction prompts, the models sometimes fail to follow the required format for travel plans. These formatting issues hinder both automatic evaluation and real-world usability, revealing limitations in instruction following and output consistency.

C Prompt Template

In this section, we present the prompts used in this study. Figures 7, 8, 9, 10, and 11 illustrate the **Next Step Prompt**, **Profile Filtering**, **Behavior Trajectory Generation**, **Quality Control**, and **B2T-Agent**, respectively.

Tool	Description	Parameter
AccommodationSearch	Accommodation options in a specified city are found, with support for preference-based filtering.	**Main Parameters:** • `city: str` • `n: int (optional)` • `preferences: dict (optional)` **Preference Keys:** • `star_rating: list of str` • `amenities: list of str`
AttractionSearch	Attractions in a specified city are found, with support for preference-based filtering.	**Main Parameters:** • `city: str` • `n: int (optional)` • `preferences: dict (optional)` **Preference Keys:** • `attraction_type: list of str` • `walk_tolerance: list of str`
EntertainmentSearch	Entertainment options in a specified city are found, with support for preference-based filtering.	**Main Parameters:** • `city: str` • `n: int (optional)` • `preferences: dict (optional)` **Preference Keys:** • `type: list of str`
FlightSearch	Flight information is retrieved, with support for preference-based filtering.	**Main Parameters:** • `origin: str` • `destination: str` • `departure_date: str` • `n: int (optional)` • `preferences: dict (optional)` **Preference Keys:** • `cabin_class: list of str`
MapGuide	Navigation routes from a departure location to a destination are searched, including walking, driving, and public transit options, either within the same city or between different cities.	• `origin_cityName: str` • `origin: str` • `destination_cityName: str` • `destination: str`
RestaurantSearch	Dining options in a specified city are explored, with support for preference-based filtering.	**Main Parameters:** • `city: str` • `n: int (optional)` • `preferences: dict (optional)` **Preference Keys:** • `cuisine: list of str` • `price_level: list of str` • `environment: list of str`
TrainSearch	Train information is retrieved.	• `origin: str` • `destination: str` • `n: int (optional)`
WeatherSearch	Weather information for a specified city and date is retrieved.	• `city: str` • `date: str (optional)` • `n: int (optional)`
Python	Python code is executed, with the ability to display outputs and handle errors.	• `code: str`

Table 7: The toolset definition comprising 9 specialized APIs. Each retrieval tool is equipped with fine-grained parameters and preference keys (e.g., *cuisine*, *star_rating*) to support precise, multi-criteria filtering of the POI database.

Category	Constraint	Description
Format (1)	JSON Format Validation	Validates overall JSON structure and required fields
Authenticity (4)	POI Authenticity	Verifies that Points of Interest exist in real city databases
	Major Transport Authenticity	Confirms availability of intercity transport like flights/trains
	Local Transport Authenticity	Ensures local transport durations are reasonable
	Business Hours Authenticity	Checks that POI visits occur within operating hours
Reasonableness (15)	Complete Meal Arrangement	Ensures three meals are arranged each day
	Non-duplicate Restaurants	No repeated restaurant recommendations across the plan
	Reasonable Meal Times	Meals occur at appropriate times of day
	Daily Accommodation Arrangement	A hotel must be arranged for each night
	Reasonable Accommodation Time	Hotel check-in should be scheduled after 18:00
	Avoid Midnight Travel	No transport activities between 23:00 and 06:00
	No Attractions After Late Major Transport	No sightseeing after arriving at a city after 18:00
	Local Transport Time Limit	Total daily local transport should not exceed 2 hours
	Weather and City Info Included	Weather and city background must be included for each destination
	At Least One Attraction	Each day must feature at least one valid attraction
	Attraction Quantity Limit	No more than 4 attractions per day
	No Time Conflicts	Activities should not overlap, with at least 30-minute gaps
	Reasonable Time Range	Activities must occur between 09:00 and 21:00
	Inter-attraction Transport Time	Transport between attractions should not exceed 1 hour
	No Long Continuous Gaps	No idle gaps longer than 2 hours during 09:00–21:00
Personalization (10)	Hotel Star Level Matching	Hotel meets user's minimum star rating
	Hotel Amenities Matching	Hotel has required facilities (e.g. WiFi, parking)
	Attraction Type Matching	Attractions align with user's preferences
	Walking Tolerance Matching	Plan respects walking endurance limitations
	Cuisine Type Matching	Meals reflect preferred cuisine types
	Price Level Matching	All items stay within specified price levels
	Dining Environment Matching	Dining settings match user preferences (e.g. quiet, lively)
	Transport Mode Matching	Intercity transport matches preferred modes
	Seat Class Matching	Matches user's preferred flight/train seat class
	Budget Constraint	Total cost stays within user-defined budget
Information Richness (2)	Information Sufficiency	Each activity includes all required fields
	Information Richness	Extra details (e.g. descriptions, tips) are included when possible

Table 8: Taxonomy of constraints derived from **real-world business scenarios**. The system enforces a total of **32** distinct constraints spanning **5** categories (e.g., Authenticity, Reasonableness) to ensure the feasibility and quality of travel plans.

Field	Data Type	Value
Historical Travel Experiences		
accommodations	array (of object)	-
restaurants	array (of object)	-
attractions	array (of object)	-
transportation	array (of object)	-
Hotel Preferences		
star_requirement	array (of string)	*2-Star & Below/Economy, 4-Star/Upscale, 3-Star/Comfort, 5-Star/Luxury, City Homestay*
essential_facilities	array (of string)	*Free WiFi, Private Bathroom, Parking, Gym, Swimming Pool, Restaurant*
Attraction Preferences		
preferred_attraction_types	array (of string)	*Park/Square, Museum/Exhibition, Cultural/Historic Site, Cityscape, Natural Landscape, Comprehensive Scenic Area, Religious Site, Theme Park, Outdoor Adventure, Water Activities, Hot Spring/Wellness, Rural/Folk Village, Animal Watching, Ice/Snow Sports, Water Conservancy Facility*
walking_tolerance	string	*Low (1-3km), Medium (3-5km), High (5km+)*
Dining Requirements		
cuisine_preferences	array (of string)	*Beverages, Snacks/Fast Food, Other Cuisines, Hot Pot, Western Food, Cantonese, Other Chinese Food, Shanghai/Zhejiang Cuisine, Sichuan Cuisine, Bakery/Desserts, Barbecue, Buffet, Northern Chinese Food, Hunan Cuisine, Seafood, Japanese/Korean Cuisine, Southeast Asian Cuisine*
price_level	string	*Budget (30-80 CNY/person), Moderate (80-150 CNY/person), Generous (150-300 CNY/person), Luxury (300-500 CNY/person)*
dining_environment	string	*Quiet, Scenic View, Family-friendly, Lively, Unique Decor*
Transportation Preferences		
intercity_transport	string	*High-speed Rail, Regular Train, Airplane, Long-distance Bus*
intracity_transport	string	*Subway, Bus, Taxi, Ride-sharing, Walking*
seat_class	string	*Economy Class, Business Class, First Class Seat, Second Class Seat*

Table 9: The schema of the user preference profile derived from **real-world travel data**. It consists of **14** fine-grained attributes across **5** dimensions (e.g., Dining, Transportation), defining both historical records and specific constraints to enable personalized planning.

Prompt Template for Next Step Prompt

```
Review the user's goals, summarize the information already collected from past conversations,
and then briefly think about what additional information is needed to create a travel guide as
detailed as the template, before selecting multiple tools to execute.
Notes:
Do not aim too high. You have multi-turn problem-solving ability, so to ensure the completeness
 and accuracy of your output, you can solve only part of the problem each time.


Keep your reasoning concise: output your reasoning in
 with no more than 100 words, focusing only on the most important information and avoiding
repetitive details.
You must use all the information obtained through tools to summarize your answer. In the
 section, do not guess dates, destination city attractions, or attraction details.
In the first reasoning step, you can only call 3 tools to determine three pieces of information
, and only after that can you execute other operations.


If the city is different from the user's current location, you need to check for round-trip
flights/trains. If the user hasn't specified a departure city, assume departure is from the
local city, in which case no train/flight information is needed.
You must end the task using <result> </result>. The <result> </result> tag must stand alone and
 cannot be combined with other tool calls.
```

Figure 7: Prompt Template for Next Step Prompt.

Prompt Template for Filtering Profile

```
Evaluate whether the following persona is reasonable:
Age: {age}
Education Background: {education}
Occupation: {occupation}
Location: {location}
Family Status: {family_status}


Sightseeing Preferences: {sightseeing_preference}
Food Preferences: {food_preference}
Hotel Preferences: {hotel_preference}
Transportation Preferences: {transportation_preference}

Evaluation criteria include but are not limited to:
1. Whether the age matches the education level (e.g., a 20-year-old PhD is unlikely; a 40-year-
old with only elementary education is rare)


2. Whether the education background aligns with the occupation (e.g., someone with only a
primary school education being a surgeon is unreasonable; a master's degree holder in a
research or senior management role makes more sense)
3. Whether the age aligns with the family status (e.g., an 18-year-old being married with
children is uncommon; a 60-year-old single person with no children may require justification)
4. Whether the occupation logically fits the location (e.g., a tech product manager is usually
based in major cities; a full-time fashion influencer in a rural town would need context)


5. Whether the income level inferred from the occupation matches the stated preferences (e.g.,
someone earning a low income preferring luxury hotels and Michelin restaurants is inconsistent)
6. Whether the transportation preferences match the age or financial condition (e.g., a 70-year
-old backpacking solo on a tight budget is uncommon; older people tend to prefer comfort and
convenience)
7. Whether sightseeing/food/hotel preferences align with the overall persona (e.g., a highly
educated researcher preferring trendy Instagram spots may seem off; a low-income person
favoring ultra-luxury hotels may be unrealistic)


8. Whether the overall persona is logically coherent (e.g., a culturally inclined person should
 show that interest in their dining, travel, or accommodation preferences)

Please respond strictly in the following JSON format and do not add any extra content:
{"reasonable": true/false, "reason": "specific reason"}

Output: {output}
```

Figure 8: Prompt Template for Filtering Profile.

Prompt Template for Behavior Trajectory Generation

```
# ROLE
You are role-playing a real user on a Chinese online travel platform (e.g.,
Ctrip / Fliggy). Your task is to generate a single-session behavior trajectory
that authentically reflects the given user profile, as if you were genuinely
browsing and exploring the platform during one visit.



# USER PROFILE
{user_profile}

# SESSION CONTEXT
Current session number: {session_id}
Previous session history (for cross-session continuity):
{previous_sessions}



# CANDIDATE POI POOL
## Preference-Matched POIs (aligned with the user's profile):
{matched_pois}
## Noise POIs (NOT aligned with the user's profile):
{noise_pois}

# ACTION SPACE
- "click"            : The user is browsing or curious. Low commitment.


- "add_to_favorites" : The user finds the POI appealing and saves it for later.
- "book"             : The user commits to the POI.

# BEHAVIORAL RULES
1. Preference Grounding: "add_to_favorites" and "book" MUST use Preference-Matched
   POIs only. Noise POIs MUST only receive "click" actions.
2. Behavioral Funnel: click -> add_to_favorites -> book. A POI must be clicked
   before it is favorited; favorited before it is booked. Most POIs stop at click.


3. Sequential Coherence:
   - Within session: consecutive actions should be topically related.
   - Cross-session: if previous sessions are provided, continue the established
     preference trajectory; do not re-book already-booked POIs; early sessions
     reflect broad exploration, later sessions narrow toward commitment.
4. Spatio-Temporal Consistency: POIs in one session should be geographically
   plausible. Decide transportation and accommodation before attractions and dining.
5. Semantic / Category Diversity: cover multiple POI categories per trajectory.


6. Noise Behavior: noise clicks should be sporadically interleaved, not clustered.
7. Realism Constraints: do NOT invent POIs outside the candidate pool. Do NOT
   mention preferences explicitly. Do NOT include reasoning or commentary.

# OUTPUT FORMAT
Return a JSON array for THIS SESSION ONLY. Each record:
[
  {


    "action_type": "click" | "add_to_favorites" | "book",
    "poi_id": "<string from candidate pool>",
    "poi_name": "<string from candidate pool>"
  },
  ...
]
Timestamps and full POI details are added automatically in post-processing.
```

Figure 9: Prompt Template for Behavior Trajectory Generation.

Prompt Template for Quality Control

```
You are a professional travel-user behavior data-quality analyst.

Your task is to evaluate the given user data. You must imitate the example below, score each of
 the three dimensions--'Completeness of Trajectory Information', 'Clarity of Preference
Reflection in the Trajectory', and 'Degree to Which the Plan Meets Constraints'--with either 0
or 1 point, and provide a structured JSON output. The output should include the scores, your


reasoning, and specific instructions that can be used to automate data modifications.

Scoring Criteria
1. Completeness of Trajectory Information
   - 1 point (Meets): The trajectory contains a clear behavior chain from browsing/exploration
to final decision (e.g., multiple behaviors such as clicks, favorites, bookings/orders) and
ends with a definitive decision point.


   - 0 points (Does Not Meet): The trajectory is overly single-patterned (e.g., only a few
clicks), lacks a final decision step, or the behavior chain is broken/illogical.

2. Clarity of Preference Reflection in the Trajectory
   - 1 point (Meets): The user's key behaviors (such as favoriting, booking) are highly
consistent with the preferences stated in persona scenic spot preference.
   - 0 points (Does Not Meet): The user's key behaviors conflict with or are completely
unrelated to the stated preferences.



3. Degree to Which the Plan Meets Constraints
   - 1 point (Meets): The generated travel plan satisfies all constraints.
   - 0 points (Does Not Meet): The generated travel plan clearly conflicts with or is
completely unrelated to the constraints.

Output Format


You must strictly output according to the JSON structure shown in the example and do not add
any explanatory text outside the JSON object.

Predefined Operation-Type Codes
- No action required: Data quality is high; no modification is needed.
- Correct user preference: When behavior conflicts with declared preferences, adjust the
preferences in the persona based on behavioral history.


- Mark incomplete trajectory: When the trajectory lacks a final decision step, mark the data so
 it can be down-weighted or filtered in downstream tasks.
- Mark for deletion: When data quality is too low to fix, recommend deleting the entire record.

Input: {input}

Output: {output}
```

Figure 10: Prompt Template for Quality Control.

Prompt Template for B2T-Agent

```
You are a seasoned travel planner with extensive experience in generating personalized plans.
Your core task is to deeply understand the user's requirements, gather information by calling
tools, and ultimately design a thorough and meticulous travel plan.
You possess an internal dictionary-structured memory module for storing and retrieving key
information. Use <memory></memory> to wrap any operation that reads or writes session memory
using "action": "read" or "write" with a "key" (and optional "value" for write). After


gathering all necessary information and crafting the complete plan, please wrap the final
itinerary in JSON format between the "<answer>" and "</answer>" tags.
Note: The delivered travel plan must be complete, covering every day of the trip.
Please note that the final output should be a JSON list (array) that includes all travel days.
Each day is an independent JSON object that follows the same internal structure.

# Tools


{"name": "AccommodationSearch", "parameters": {"type": "object", "properties": {"city": {"title
": "City", "type": "string"}, "n": {"default": 5, "title": "N", "type": "integer"}, "
preferences": {"additionalProperties": true, "default": null, "title": "Preferences", "type": "
object"}}, "required": ["city"]}}
{"name": "RestaurantSearch", "parameters": {"type": "object", "properties": {"city": {"title":
"City", "type": "string"}, "n": {"default": 5, "title": "N", "type": "integer"}, "preferences":
 {"additionalProperties": true, "default": null, "title": "Preferences", "type": "object"}}, "
required": ["city"]}}


{"name": "FlightSearch", "parameters": {"type": "object", "properties": {"origin": {"title": "
Origin", "type": "string"}, "destination": {"title": "Destination", "type": "string"}, "
departure_date": {"title": "Departure Date", "type": "string"}, "n": {"default": 5, "title": "N
", "type": "integer"}, "preferences": {"additionalProperties": true, "default": null, "title":
"Preferences", "type": "object"}}, "required": ["origin", "destination", "departure_date"]}}
{"name": "TrainSearch", "parameters": {"type": "object", "properties": {"origin": {"title": "
Origin", "type": "string"}, "destination": {"title": "Destination", "type": "string"}, "n": {"
default": 5, "title": "N", "type": "integer"}}, "required": ["origin", "destination"]}}


{"name": "WeatherSearch", "parameters": {"type": "object", "properties": {"city": {"title": "
City", "type": "string"}, "date": {"default": null, "title": "Date", "type": "string"}, "n": {"
default": 5, "title": "N", "type": "integer"}}, "required": ["city"]}}
{"name": "EntertainmentSearch", "parameters": {"type": "object", "properties": {"city": {"title
": "City", "type": "string"}, "n": {"default": 5, "title": "N", "type": "integer"}, "
preferences": {"additionalProperties": true, "default": null, "title": "Preferences", "type": "
object"}}, "required": ["city"]}}


{"name": "AttractionSearch", "parameters": {"type": "object", "properties": {"city": {"title":
"City", "type": "string"}, "n": {"default": 5, "title": "N", "type": "integer"}, "preferences":
 {"additionalProperties": true, "default": null, "title": "Preferences", "type": "object"}}, "
required": ["city"]}}
{"name": "MapGuide", "parameters": {"type": "object", "properties": {"origin_cityName": {"title
": "Origin Cityname", "type": "string"}, "origin": {"title": "Origin", "type": "string"}, "
destination_cityName": {"title": "destination_cityname", "type": "string"}, "destination": {"
title": "Destination", "type": "string"}}, "required": ["origin_cityName", "origin", "


destination_cityName", "destination"]}}

Behavior trajectory: {behavior trajectory}
Instruction: {instruction}
```

Figure 11: Prompt Template for B2T-Agent.