ChinaTravel: A Real-World Benchmark for Language Agents in Chinese Travel Planning

Anonymous ACL submission

Abstract

Recent advances in LLMs, particularly in language reasoning and tool integration, have rapidly sparked the real-world development of *Language Agents*. Among these, travel planning represents a prominent domain, combing complex multi-objective planning challenges with practical deployment demands. Existing benchmarks, however, often oversimplify real-world requirements by focusing on synthetic queries and limited constraints. To address this gap, we introduce *ChinaTravel*, the first benchmark designed for authentic Chinese travel planning scenarios. We collect the travel requirements from questionnaires and propose a compositionally generalizable domain-specific language that enables a scalable evaluation process, covering feasibility, constraint satisfaction, and preference comparison. Empirical studies reveal the potential of neuro-symbolic agents in travel planning, achieving 27.9% constraint satisfaction rate on human queries, a 10.7× improvement over purely-neural models (2.6%). Moreover, we identify key challenges in real-world deployments, including open language reasoning and unseen concept composition. These findings highlight the significance of ChinaTravel as a pivotal milestone for advancing language agents in complex, real-world planning scenarios.

1 Introduction

A long-standing goal in AI is to build planning agents that are reliable and general, able to assist humans in real-world tasks. Recently, Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023) have demonstrated remarkable potential in achieving human-level understanding and reasoning capabilities. This has sparked the rapid development of a field called *Language Agents*, employing LLMs to perceive the surroundings, reason the solutions, and take appropriate actions, ultimately building an autonomous planning agent (Shinn et al., 2024; Yao et al., 2023; Xi et al., 2023; Jimenez et al., 2024).

Among numerous real-world planning tasks, travel planning stands out as a significant domain, presenting both academic challenges and practical value due to its inherent complexity and real-world relevance. Specifically, given a query, travel planning agents require information integration from various tools (e.g., searching for flights, restaurants, and hotels) to generate a feasible itinerary. This involves making interdependent decisions across multiple aspects such as spatial, temporal, and financial dimensions, all while meeting the user's requirements and preferences (e.g., budget, dining habits, etc). This travel planning task presents both significant practical value and important research challenges. As a pervasive yet complex activity, it demands considerable time investment, creating compelling need for AI assistance. Academically, it constitutes a long-horizon planning objective that involves various hard and soft constraints, posing unique challenges for planning agents.

To evaluate the existing language agents on travel planning tasks, TravelPlanner (Xie et al., 2024) first provides a benchmark. However, the TravelPlanner benchmark focuses solely on U.S. domestic travel, consisting only of intercity itineraries, which are less common in real-world travel planning scenarios (as it is more typical to travel within a single city for several days). Additionally, it only includes synthesized queries, lacking real human travel queries. Furthermore, just a few months after the benchmark's release, Hao et al. (2024) proposed a neural-symbolic solution that integrates formal verification tools into language agents, achieving a 97% success rate. This underscores the oversimplification of the TravelPlanner benchmark. Therefore, it is highly desirable to develop a novel benchmark that better reflects real human travel habits and requirements, while also capturing the complexity of the task.

In this paper, we provide ChinaTravel, a novel travel planning benchmark, tailored to authentic Chinese travel requirements, concentrating on multi-

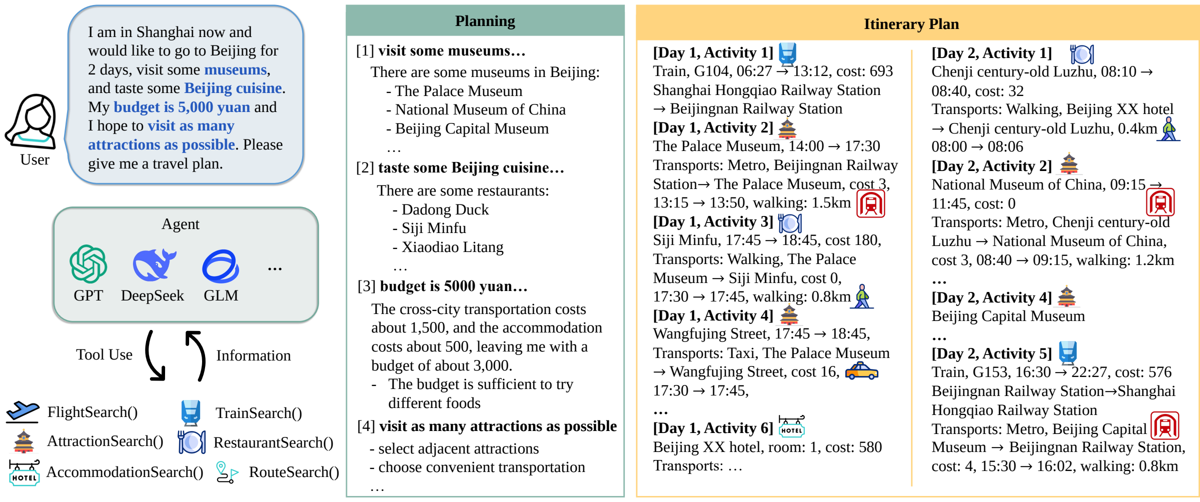


Figure 1: Overview of ChinaTravel. Given a query, language agents employ various tools to gather information and plan a multi-day multi-POI itinerary. The agents are expected to provide a feasible and reasonable plan while satisfying the hard logical constraints and soft preference requirements. To provide convenience for global researchers, we provide an English translation of the original Chinese information here.

point-of-interest (multi-POI) itineraries (as illustrated in Fig. 1). Compared to TravelPlanner, ChinaTravel is more realistic and challenging. The main contributions are summarized as follows.

- **Comprehensive Evaluation Framework:** ChinaTravel provides a rich sandbox with authentic travel data, a domain-specific language for scalable requirements definition and automated evaluation, and diverse metrics covering feasibility, constraint satisfaction, and preference ranking.
- **Integration of Synthetic and Human Queries:** ChinaTravel includes both LLM-generated and human-derived queries, offering a realistic and open testbed for evaluating agents in addressing authentic and multifaceted travel requirements.
- **Empirical Neuro-Symbolic Insights:** Extensive experiments are conducted and the results reveal that neuro-symbolic agents significantly outperform pure LLM-based solutions, achieving a constraint satisfaction rate of 27.9% compared to 2.60% by purely neural methods, thus highlighting their promise for travel planning tasks.
- **Identified Challenges for Future Research:** We pinpoint key challenges of open-world requirements: open language reasoning, and unseen concept composition, providing a foundation for advancing agents toward real-world applicability.

Overall, ChinaTravel provides a challenging yet meaningful testbed for evaluating language agents in travel planning, serving as a critical bridge between academic research and practical applications.

2 ChinaTravel Benchmark

Motivated by China's substantial travel demand, ChinaTravel provides a sandbox environment for generating multi-day itineraries with multiple POIs across specified cities. This benchmark is meticulously designed to provide a comprehensive and scalable evaluation framework for language agents in travel planning, encompassing three critical dimensions: environmental feasibility, constraint satisfaction, and preference comparison.

2.1 Environment Information

ChinaTravel provides a sandbox with real-world travel information. We collect information from 10 of the most popular cities in China. It includes 720 airplanes and 5,770 trains connecting these cities, with records detailing departure and arrival times, origins, destinations, and ticket prices. Additionally, the dataset contains 3,413 attractions, 4,655 restaurants, and 4,124 hotels, each annotated with name, location, opening hours, and per-person prices. Type annotations for these POIs are included to meet user needs. Fig. 2 has provided an illustration of the collected information from Beijing and Nanjing, two of the most popular cities in China. For a more realistic interaction, we simulate the API interface of real market applications to query real-time information. The detailed designs of the sandbox are available in App. B.1. Environmental constraints act as a feasibility metric, ensuring that the generated plans are both valid and effective. For example, POIs in the plan must exist in the designated city, transportation options must be viable,

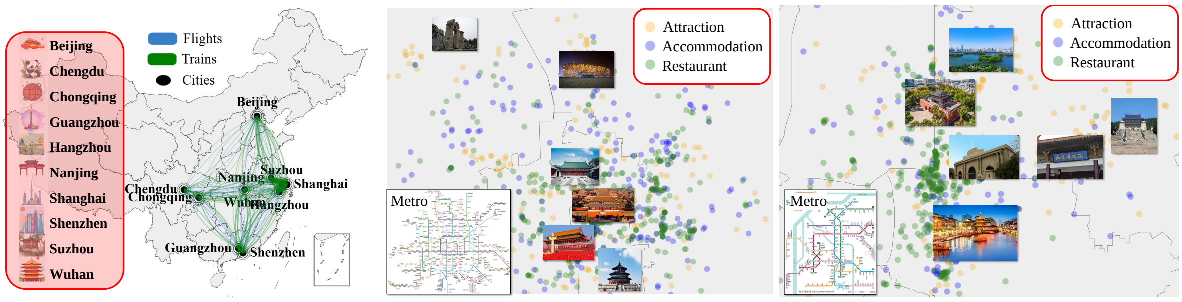


Figure 2: Illustration of **ChinaTravel Sandbox Environment**. Our sandbox incorporates travel information from 10 of the most popular cities in China, offering comprehensive information on attractions, accommodations, and restaurants essential for travel planning. Here is the visualization of information from Beijing and Nanjing.

Evaluation Metrics	**Environment Constraints**
Cross-city Transportation	Available Trains or Airplanes across cities.
	Correct information of cost and schedule.
Inner-city Transportation	Available Metro, Taxi or Walking between different positions.
	Correct information of cost, distance and duration
Attractions	Available Attractions in the target city, visiting in their open time.
	Attraction choices should not be repeated throughout the trip.
	Correct information of cost.
Restaurants	Available Restruants in the target city, visiting in their open time.
	Restaurant choices should not be repeated throughout the trip.
	Breakfast, lunch, and dinner are served at their designated meal times.
	Correct information of cost.
Accommodation	Available Accommodation in the target city.
	Room information to meet headcounts.
Time	The given activity events occur in chronological order.
Space	Events at different positions should provide transport information.

Table 1: Descriptions of **Environment Constraints** for two benchmarks. Constraints in black are common in both TravelPlanner and ChinaTravel. Metrics in brown are the metrics only in our benchmark.

and time information must remain accurate. Tab. 1 summarizes the environmental constraints.

2.2 Logical Constraint

A crucial ability for travel planning is to effectively satisfy personalized user needs. We extend the form of logical constraints from TravelPlanner (Xie et al., 2024) and present a Domain-Specific Language (DSL) to support general compositional reasoning in logical constraints. ChinaTravel's DSL is a general set of pre-defined concept functions with built-in implementations and is listed in Tab. 2. TravelPlanner relies on 5 pre-defined concepts {total budget, room rules, room types, cuisines, and transportation types}, to evaluate the logical constraints, where each concept is equivalent to a specific logical requirement. We find this design limits the ability to validate diverse logical needs in an open-world context. For example, such an evaluation cannot express that the dining expenses should be within 1000 CNY or that arriving in Shanghai should be before 6 PM on the second day, despite the generated plan already including the expenses and time information of each activity. Each new logical requirement necessitates human intervention for incremental definition. To address this issue, our approach is grounded in a DSL-based solution that leverages basic concept functions and syntax to express and fulfill various logical requirements.

```
# Dining expenses <= 1000 CNY.
dining_cost = 0
for act_i in allactivities(plan):
  typ = activity_type(act_i)
  if typ=="breakfast" or typ=="lunch" or
     typ=="dinner": dining_cost =
    dining_cost + activity_cost(act_i)
return dining_cost <= 1000
```

Name	Syntax	Description
variables	$x, y, z, \cdots$	Variables that refer to activities in the travel planning domain.
not	$not\ expr$	The negation of an Boolean-valued expression.
and,or	$expr_1\ and\ expr_2$	The conjunction/disjunction of an Boolean-valued expression.
$<, >, ==$	$expr_1 < expr_2$	Return an expression with built-in number comparison functions.
$+, -, *, /$	$expr_1 + expr_2$	Return an expression with built-in number calculation functions.
attributes	$cost(var)$	A function that takes activities as inputs and returns the attributes, such as cost, type or time.
relation	$dist(expr_1, expr_2)$	A function that takes locations as inputs and returns the distance.
effect	$var = expr$	An assignment affects a variable var with the expression $expr$.
union, inter, diff	$uni(\{var\}_1, \{var\}_2)$	Return a set with the built-in union/intersection/difference operations of given two sets.
enumerate	*for var in* $\{var\}$	Enumerate all variables in the collection $\{var\}$.
when	*if* $expr$: *effect*	The conditional effect takes a Boolean-valued condition of the expression $expr$, and the effect *effect*.

Table 2: ChinaTravel's Domain-Specific Language (DSL) for logical constraints.

```
# Arriving in Shanghai should be before
    6 PM on the second day.
return_time = 0
for act_i in day_activities(plan, 2):
  typ = activity_type(act_i)
  dest = transport_destination(act_i)
  if (typ=="train" or typ=="airplane")
      and des=="Shanghai": return_time
      == activity_endtime(act_i)
return return_time < "18:00"
```

The DSL can represent varying requirements through concept composition in a Python format, and perform automated validation of plans using a Python compiler. This strategy maximizes the evaluation capability of the ChinaTravel benchmark. The App. B.2 provides a detailed tutorial on DSL expression with more practical examples.

2.3 Preference Requirement

Travel requirements encompass not only hard logical constraints but also soft preferences. The term "soft" implies that these preferences cannot be addressed as boolean constraint satisfaction problems, instead, they involve quantitative comparisons based on continuous values. This distinction highlights the unique nature of preference-based requirements compared to logical constraints. Common preferences identified through surveys include maximizing the number of attractions visited, minimizing transport time between POIs, and visiting positions near the specific POI, among others. In ChinaTravel, we formalize such preferences as minimization or maximization objectives via our DSL, thereby providing an automated evaluation.

```
# The number of attractions visited
count = 0
for act_i in all_activities(plan):
  if activity_type(act_i)=="attraction":
      count = count + 1
return count
```

2.4 Benchmark Construction

ChinaTravel provides user queries reflecting diverse requirements through a four-stage process that integrates LLM-based generation with questionnaires.

Stage I: Manual design of database and APIs. We collect travel information for multi-day, multi-POI itineraries across attractions, accommodations, and transportation. We define essential POI features, such as cuisine types and hotel characteristics, to construct the database from public information. APIs are designed to support agent queries via regular expressions and modeled after commercial APIs to ensure realism. See App. B.1 for details.

Stage II: Automatic data generation with LLMs. We define common travel information (e.g., origin, destination, days, number of people) and logical constraints to model travel tasks. To enable scalable queries, query skeletons are randomly constructed from this information and transformed into natural language queries using an advanced LLM, DeepSeek-V2.5, which is selected for its strong Chinese language proficiency, robust instruction-following capabilities, and cost efficiency. The generated queries are categorized into two difficulty levels: *Easy*, with 1 logical requirement beyond basic constraints like people number and trip duration, and *Medium*, with 3–5 additional logical requirements. We encourage the LLM to generate diverse, human-like expressions, such as turning "Taste Beijing cuisine" into "Try local food in Beijing." See App. B.3 for more details about the synthesis.

Stage III: Quality control and auto-validation. To ensure data quality, we manually check whether the generated queries conform to symbolic skeletons, and re-calibrate natural language descriptions that contain ambiguities. Based on the symbolic skeletons of queries, we could verify whether the plan can pass the required logical constraints by executing the DSL code via Python compiler. Building on this, we ensure that each query has at least one solution that satisfies the logical constraints by implementing a heuristic search algorithm.

Stage IV: Open requirements from humans. After the first round of closed-loop development with LLMs, including data generation and annotation, baseline development, and evaluation, we further collected travel requirements from more than 250 humans through questionnaires. Based on a new round of quality control on these data, a more challenging set with 154 queries is constructed. These queries even include unseen logical constraints in the deployment process, such as 'departure time' and 'dining cost', reflecting the real challenges of the travel planning system. We carefully annotate the required logical constraints for each query based on the DSL, enabling the automated evaluation of these challenging samples and forming the *Human* level dataset.

To support global research on travel planning, we provide an English version of all queries in ChinaTravel. However, we recommend that researchers primarily use the Chinese version, as it better captures the expression from native speakers.

3 Empirical Study

LLMs. We test both state-of-the-art proprietary and open LLMs: OpenAI GPT-4o, DeepSeek-V2.5, as well as Qwen-2.5-7B (Bai et al., 2023). The first two models are chosen for their strong performance, while the latter is selected for their Chinese language capabilities and ability to perform inference with limited local computational resources.

Metrics. We examine the Delivery Rate (DR), Environmental Pass Rate (CPR), Logical Pass Rate (LPR), and Final Pass Rate (FPR) from TravelPlanner (Xie et al., 2024). Furthermore, we design a novel metric, Conditional Logical Pass Rate (C-LPR), evaluating the success rate of plans that first fulfill environmental constraints prior to logical constraints. It ensures that logical requirements are met within a realistic travel context, eliminating cases where unrealistic or incorrect information might lead to shortcutting logical constraints, such as misreporting costs to fit budget requirements. By introducing C-LPR, we aim to enhance the feasibility and meaningfulness of constraint satisfaction.

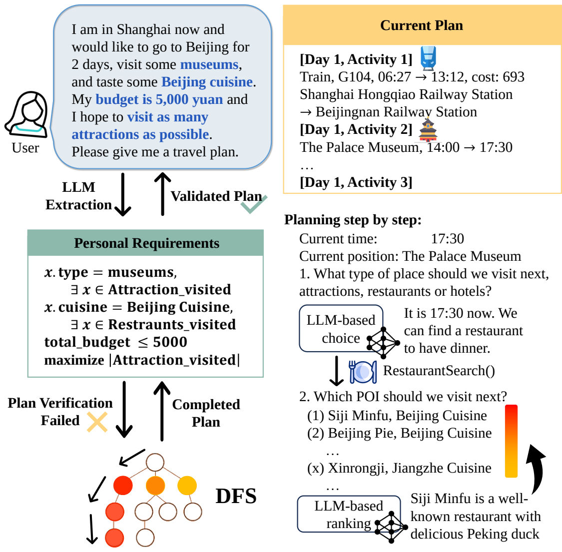


Figure 3: NeSy Planning with depth-first-search solver.

$$C\text{-}LPR = \frac{\sum_{p \in P} \mathbb{1}_{passed(Env,p)} \cdot \sum_{c \in C_p} \mathbb{1}_{passed(c,p)}}{\sum_{p \in P} |C_p|}$$

P is the plan set, C_p is the set of constraints for plan p, and passed(c, p) indicates whether p satisfies c.

Methods. We evaluate the performance of both pure-LLM-based and neuro-symbolic solutions on the ChinaTravel benchmark. For the former, we primarily test the well-known method, ReAct (Yao et al., 2023), and its Act-only ablation. We exclude Reflexion (Shinn et al., 2024) due to its performance being similar to ReAct on the TravelPlanner (Xie et al., 2024) and the high economic overhead associated with the larger input token size. For the latter, we adapt existing neuro-symbolic pipelines (Hao et al., 2024; Pan et al., 2023; Deng et al., 2024) using our proposed DSL to handle the complexities of multi-day, multi-POI itineraries.

3.1 Neuro-Symbolic Planning

This subsection presents a neuro-symbolic solution as a preliminary baseline for ChinaTravel. This

	LLMs	DR	EPR		LPR		C-LPR	FPR
			Micro	Macro	Micro	Macro		
***Easy* (#300)**								
Act		70.4	49.9	0	64.6	30.8	0	0
		97.5	70.8	0	86.8	68.8	0	0
ReAct		43.3	40.8	0	41.9	19.6	0	0
		95.4	48.2	0	71.3	32.9	0	0
ReAct (one-shot)		77.5	68.3	6.25	74.1	52.5	5.77	5.42
		94.2	68.1	0	**89.4**	**70.8**	0	0
NeSy Planning		78.6	**75.9**	50.6	79.7	64.6	48.6	48.0
		75.0	73.6	**64.0**	73.5	63.3	**61.7**	**60.6**
		72.3	67.0	34.0	70.4	49.6	32.6	28.3
NeSy Planning* (Oracle Translation)		**82.6**	**81.7**	**75.0**	**82.2**	**75.3**	**75.0**	**74.0**
		66.6	66.7	66.0	64.6	63.6	64.6	62.6
		69.3	69.3	59.3	70.2	59.6	59.3	57.9
***Medium* (#150)**								
Act		72.7	52.3	0	63.5	15.3	0	0
		97.4	70.5	0	89.3	55.3	0	0
ReAct		41.3	35.2	0	37.6	4.0	0	0
		92.0	54.8	0	78.6	22.7	0	0
ReAct (one-shot)		82.7	**77.1**	3.33	82.6	48.7	2.95	1.33
		94.7	69.2	0.67	**91.8**	**64.0**	0.53	0
NeSy Planning		71.3	71.9	**69.3**	69.4	50.0	**69.3**	**46.7**
		68.0	68.0	68.0	64.1	46.6	64.1	**46.7**
		53.3	45.9	16.0	49.2	33.3	14.8	8.50
NeSy Planning* (Oracle Translation)		**68.6**	**65.4**	54.0	**66.2**	**61.3**	52.5	54.0
		60.8	59.4	**54.9**	60.3	58.2	**60.3**	**56.9**
		53.3	51.3	36.6	51.9	43.3	34.8	34.6
***Human* (#154)**								
ReAct		36.4	29.5	0.65	35.2	16.2	0.38	0
		96.1	50.5	0	**72.4**	32.5	0	0
ReAct (one-shot)		55.2	**57.3**	2.60	64.6	44.2	1.71	2.60
		69.5	46.3	0	63.6	46.8	0	0
NeSy Planning		45.4	46.6	40.9	40.9	**33.1**	35.3	**27.9**
		45.4	50.1	**45.4**	40.9	29.8	**38.5**	**27.9**
		42.8	47.4	42.2	36.2	27.2	34.4	25.3
NeSy Planning* (Oracle Translation)		50.6	**48.9**	36.3	45.9	40.2	32.0	35.0
		52.6	46.9	**42.9**	**47.6**	**40.9**	**43.9**	**40.9**
		41.5	41.1	31.1	36.5	33.7	25.0	28.5

Table 3: Main results of different LLMs and planning strategies on the ChinaTravel benchmark. LLMs: : DeepSeek-V2.5, : GPT-4o-2024-08-06, :Qwen2.5-7B.

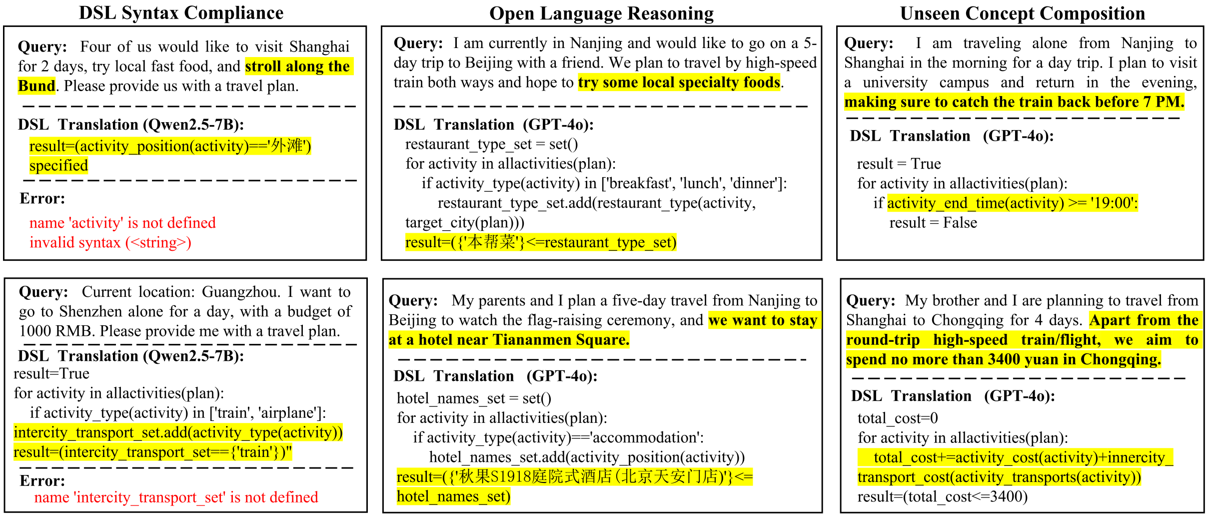


Figure 4: Challenges in the Neuro-Symbolic Planning.

solution consists of two stages. Stage 1: *NL2DSL translation* translates natural language queries into logical, preference-based DSL requirements. We use Reflexion (Shinn et al., 2024) and a DSL syntax checker to iteratively assist the LLMs (5 rounds in experiments). Stage 2: *Interactive search* uses a neuro-symbolic solver to sequentially arrange activities, guided by a symbolic sketch and LLM-driven POI recommendations, generating a multi-day itinerary with DSL validation. If constraints are violated, the process backtracks until a feasible solution is found. To ensure fairness, the symbolic sketch search is limited to 5 minutes per query, excluding LLM inference time. To observe the performance across the two stages, we also evaluated the planning results based on the Oracle DSL. App. D includes pseudo-code and LLM prompts.

3.2 Main Results

Based on the results presented in Table 3, we have the following observations and analyses:

Pure LLMs struggle in ChinaTravel. The DR evaluates an agent's ability to generate valid JSON plans (see Fig. 1). While high DRs indicate that advanced LLMs can produce structured outputs for travel planning, the near-zero EPR (Environmental Constraints Pass Rate) reveals their inability to gather and strictly adhere to required information. The sole exception is the DeepSeek model, which achieves the 6% EPR and 5% FPR at easy level, likely due to its strong capability to follow Chinese requirements. ReAct (one-shot, GPT-4o) excels in Macro LPR but achieves no FPR, suggesting it circumvents constraints via shortcuts. Our proposed C-LPR metric offers a more reliable measure of logical constraints, serving as a supplement to FPR.

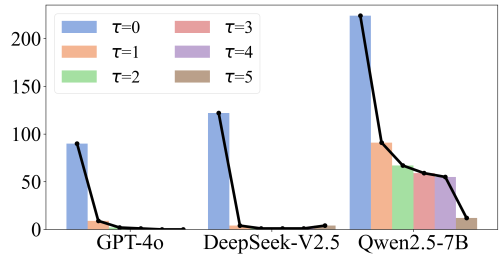


Figure 5: Syntax errors across reflexion rounds τ.

Nesy Planning provides a promising solution. Our NeSy Planning framework integrates symbolic programs to orchestrate travel planning and tool management while utilizing LLMs to extract language-based requirements and prioritize POIs. By separating planning (flexible natural language handling) from grounding (precise execution), the framework enhances adaptability and ensures compliance with constraints. Across all data subsets, NeSy methods outperform pure-LLM approaches. With GPT-4o as the backend, it achieves FPRs of 60.6%, 46.7%, and 27.9% on three subsets, highlighting the effectiveness of NeSy solutions for travel planning with complex constraints.

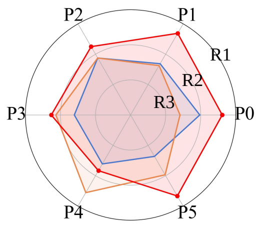
Challenges Persist for Nesy Planning. The performance gap between standard and oracle modes underscores the importance of DSL translation in NeSy planning. Inadequate translations may result in plan searches failing to meet user requirements, while incorrect translations can misguide the search, making feasible solutions unattainable. Among the three LLMs, GPT-4o performs the best, with minimal gaps between modes, indicating its relatively accurate DSL generation effectively supports the search process. We conclude with three challenges and provide the corresponding cases in the Fig. 4.

(1) DSL Syntax Compliance: As shown in Fig. 5, while the reflexion process with syntax checker significantly reduces syntax errors, the Qwen-7B model demonstrates weaker compliance than GPT-4o and DeepSeek, directly resulting in its lower performance in the Tab. 3. **(2) Open Language Reasoning**: Although GPT-4o exhibits relatively fewer syntax errors in translation, it still struggles with context-dependent meanings. For instance, when a user requests 本地菜 (local cuisine), GPT-4o maps it to 本帮菜, ignoring the logical connection that in Beijing, it should align with 北京菜 (Beijing cuisine). **(3) Unseen Concept Composition**: Real-world requirements derived from human data are inherently diverse and complex, making expecting models to encounter all possible needs during development impractical. A more feasible way is to emulate human reasoning by generalizing existing knowledge to novel problems. Based on our DSL design, LLMs can express new logical requirements through combinations of concept functions. However, compositional generalization remains a challenge. GPT-4o misinterpreted a time constraint '晚上7点前赶上回程的火车' as applying to all activities instead of correctly limiting only the return train's departure time to before 19:00.

In summary, ChinaTravel poses significant challenges for current agents. Neuro-symbolic agents outperform pure-LLM approaches in constraint satisfaction, showing strong potential for real-world travel planning. With realistic queries and a versatile DSL for constraint validation, we highlight the critical challenges while providing a foundation for advancing neuro-symbolic systems in practice.

3.3 Ablation Study with Preference

The comparison of preferences should be conducted under the premise that both environmental and logical constraints are satisfied. Given the limited FPR achieved by existing methods on the challenging ChinaTravel, we perform a separate analysis of preference optimization in this section. Specifically, we sampled 50 queries from the easy subset that NeSy-DeepSeek-Oracle successfully passed as seed samples. Based on these, six subsets were created by introducing common preferences identified from user surveys. Three comparative scenarios were designed to explore the roles of LLMs and symbolic search in optimizing preferences during NeSy Planning: (1) Baseline Query (BQ): Results obtained by directly querying the seed samples without preference requirements. (2) Preference-Enhanced Query



NeSy Planning	BQ	PEQ	PDS
Daily attractions ↑	0.75	0.79	**1.63**
Transport time ↓	27.0	26.9	**24.8**
Transport time to rest. ↓	**22.2**	26.8	22.8
Food cost ratio ↑	0.19	0.29	**0.32**
Hotel cost ↓	1350	559	**519**
Distance to POI ↓	30.9	30.6	**26.1**

Figure 6: Ablation on preference ranking.

(PEQ): Results based on seed samples augmented with natural language preference expressions (e.g., "visit more attractions"), evaluating whether embedding preferences into POI recommendations via LLMs improves outcomes. (3) Preference-Driven Search (PDS): Results using both natural language and DSL-based expressions, where the agent, within the 5-minute search time limit, computes the preference concept for solutions that pass environmental and logical constraints and retains plans that maximize or minimize the preference objective. The results are provided in Fig. 6 (where ↑ /↓ indicate maximization/minimization).

We found that PEQ outperforms BQ in preference optimization. This ablation demonstrates that LLMs can effectively capture natural language needs during the POI ranking stage, contributing to preference improvements. However, on P2, PEQ underperforms BQ, indicating that LLMs can sometimes have a negative impact. This may be due to the complexity of the preference in P2, which involves minimizing transport time to restaurants, leading to misinterpretation. PDS achieves more significant improvements in preference optimization, relying on DSL-based preference calculations that filter plans more effectively over extended search times. This supports the scalability of DSL in preference optimization but also highlights the pressing need for more efficient algorithms.

4 Conclusion

We present ChinaTravel, a benchmark for multi-day multi-POI travel planning focused on authentic Chinese needs. We address the limitations of previous benchmarks by incorporating open-ended and diverse human queries, capturing real-world user needs. Additionally, we propose a scalable evaluation framework based on DSL, enabling comprehensive assessments of feasibility, constraint satisfaction, and preference comparison. These advancements provide a foundation for developing language agents capable of meeting diverse user requirements and delivering reliable travel solutions.

5 Limitations

Our research represents a significant step forward in evaluating the travel planning capabilities of language agents, but it is not without challenges. One limitation lies in its focus on Chinese travel planning. Due to the inherent differences in natural language, the translated versions of queries may fail to fully capture the challenges of understanding requirements in Chinese queries, potentially limiting its applicability in a global context. However, given the substantial demand within China's travel market, we believe a benchmark tailored to Chinese travel planning is both necessary and socially valuable. Although our benchmark is comprehensive, it may not encompass the full range of requirements encountered in real-world scenarios. The high cost of collecting authentic data has limited the number of human queries in our study. To address this, future work will focus on combining LLMs with real user queries to automate the generation of a wider variety of human-like queries. Continuous refinement and expansion of our benchmark are crucial for more accurately reflecting the realistic travel planning needs.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. 2002. Deep blue. *Artificial intelligence*, 134(1-2):57–83.

Aili Chen, Xuyang Ge, Ziquan Fu, Yanghua Xiao, and Jiangjie Chen. 2024. TravelAgent: An AI assistant for personalized travel planning. *arXiv preprint arXiv:2409.08069*.

Wang-Zhou Dai, Qiu-Ling Xu, Yang Yu, and Zhi-Hua Zhou. 2019. Bridging machine learning and logical reasoning by abductive learning. In *Advances in Neural Information Processing Systems*, pages 2811–2822.

Shujie Deng, Honghua Dong, and Xujie Si. 2024. Enhancing and evaluating logical reasoning abilities of large language models. In *Proceedings of the ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962.

Sajal Halder, Kwan Hui Lim, Jeffrey Chan, and Xiuzhen Zhang. 2024. A survey on personalized itinerary recommendation: From optimisation to deep learning. *Applied Soft Computing*, 152:111200.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2024. Large language models can plan your travels rigorously with formal verification tools. *CoRR*, abs/2404.11891.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *Proceedings of the 12th International Conference on Learning Representations*.

Weiyu Liu, Geng Chen, Joy Hsu, Jiayuan Mao, and Jiajun Wu. 2024. Learning planning abstractions from language. In *Proceedings of the 12th International Conference on Learning Representations*.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3753–3763.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 3806–3824.

Vibhor Sharma, Monika Goyal, and Drishti Malik. 2017. An intelligent behaviour shown by chatbot system. *International Journal of New Technology and Research*, 3(4):263312.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Kebing Hou, Dingyi Zhuang, Xiaotong Guo, Jinhua Zhao, Zhan Zhao, and Wei Ma. 2024. Synergizing spatial optimization with large language models for open-domain urban itinerary planning. *CoRR*, abs/2402.07204.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6545–6554.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

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*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Guiming Chen, Jianquan Li, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. Huatuogpt, towards taming language model to be a doctor. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 10859–10885.

A Discussion with Related Work

LLM-based Agents have demonstrated significant capability in understanding complex instructions and employing domain-specific tools to complete tasks, showcasing their potential in fields such as visual reasoning (Gupta and Kembhavi, 2023), healthcare (Zhang et al., 2023) and robotics (Liu et al., 2024). This reduces the reliance of previous agents on domain-specific efforts, that is, either mainly following domain-specific rules to plan (rule-based agents, such as DeepBlue (Campbell et al., 2002) and Eliza (Sharma et al., 2017)) or mainly learning from domain-specific data to plan (reinforcement-learning-based agents, such as AlphaGo (Silver et al., 2017) and Atari DQN (Mnih et al., 2013)). While the language agents have shown promising results in some domains, most of their planning scenarios are limited to simple tasks with single objective function and fail in the travel planning benchmark with complex logical constraints.

Neuro-Symbolic Learning explores to combine traditional symbolic reasoning with learning to enhance the reliability (Manhaeve et al., 2018; Wang et al., 2019; Dai et al., 2019). In the era of large language models, Pan et al. (2023) presents the LogicLM integrates LLMs with separate symbolic solvers for various logical reasoning tasks. They first utilize LLMs to translate a natural language problem into a symbolic formulation. Afterward, a deterministic symbolic solver performs inference on the formulated problem to ensure the correctness of the results. Deng et al. (2024) supplement LogicLM with a Self-Refinement Module to enhance the reliability of LLM translation. In the travel planning domain, Hao et al. (2024) presents a framework with a similar pipeline. It first extracts the logical constraints from natural language queries and then formalizes them into SMT code.

Thanks to SMT solvers being sound and complete, this neuro-symbolic solution guarantees the generated plans are correct and has basically solved the TravelPlanner benchmark with a 97% pass rate.

Travel Planning is a time-consuming task even for humans, encompassing travel-related information gathering, POI selection, route mapping, and customization to meet diverse user needs (Halder et al., 2024). Natural languages are one of the most common ways for users to express their travel requirements. However, the ambiguity and complexity of travel requirements make it still challenging for LLMs to generate accurate and reliable travel plans. Xie et al. (2024) presents the TravelPlanner benchmark for cross-city travel planning and reveals the inadequacies of pure-LLM-driven agents. TravelPlanner generates user queries through LLMs and provides a rigorous evaluation mechanism to verify whether the provided plans can meet the logical constraints in the queries. It has become a pivotal benchmark for language agents in real-world travel planning. Tang et al. (2024) study the open-domain urban itinerary planning where a single-day multi-POI plan is required. They integrates spatial optimization with large language models and present a system ITTNERA, to provide customized urban itineraries based on user needs. A concurrent work, TravelAgent (Chen et al., 2024), also considers a multi-day multi-POI travel planning problem for the specified city. It constructs an LLM-powered system to provide personalized plans. However, due to the high cost of collecting and annotating real travel needs, they evaluate the proposed TravelAgent in only 20 queries. This also demonstrates the necessity of introducing a new benchmark for travel planning.

B Detailed Design of ChinaTravel

B.1 Sandbox Information

We started collecting travel information with the motivation of planning a multi-day, multi-POI itinerary in four aspects: attractions, accommodation, activities, and transportation. Developers first determine the POI description information that needs to be obtained from the user's perspective, such as cuisine and hotel features. Based on this feature set, we collect public information to construct the database. For the design of APIs, we directly support queries based on the regular expressions from agents. At the same time, we expect the design of APIs to have similar features and characteristics to existing commercial APIs, enabling our dataset to be applicable to more realistic scenarios. The information our database contains is shown in Table 4 and the APIs we offer is in Table 5

B.2 Tutorial of DSL Expression

Here is a tutorial, that provides a step-by-step guide to utilizing ChinaTravel's Domain-Specific Language (DSL) with predefined concept functions for expressing logical constraints and preferences.

DSL Overview In the main body of this paper, we have detailed the basics of our DSL in the Table 2. The DSL is a Python-like language designed to formalize travel planning requirements into executable code. It enables automated validation of itineraries against user constraints and preferences. Key components include: 1) *Concept Functions*: Predefined functions (e.g., activity_cost, poi_distance) that extract attributes from travel plans. 2) *Operators*: Logical (and, or, not), arithmetic (+, -, *, /), and comparison operators (<, >, ==). 3) *Control Structures*: Loops (for), conditionals (if), and set operations (union, intersection).

Core Concept Functions We have defined 35 concept functions. Their definition and implementation is in Table 8, 9, 10 and 11. Below are common use cases:

Example: Budget Constraint User Query: "Total expenses must not exceed 5000 CNY."

```
total_cost = 0
for act in all_activities(plan):
    total_cost += activity_cost(act)
    total_cost +=
        innercity_transport_cost(
        activity_transports(act))
return total_cost <= 5000
```

The function all_activities(plan) iterates through all activities in the itinerary. The function activity_cost retrieves the cost of each activity. The function innercity_transport_cost sums transportation expenses. Based on Python syntax, combining these concept functions can calculate the cost of the entire plan, thereby determining whether the budget constraints are met.

Debugging Tips (1) Syntax Validation: Use the python compiler to check for syntax errors (e.g., missing colons, undefined variables). (2) Unit Testing: Test individual concept functions (e.g., poi_distance) with mock itineraries. (3) Iterative Refinement: For ambiguous requirements (e.g., "local cuisine"), map natural language to precise DSL

Tool	Information
Attractions	Name, Type, Latitude, Longitude, Opentime, Endtime, Price, Recommendmintime, Recommendmaxtime
Accommodations	Name, Name_en, Featurehoteltype, Latitude, Longitude, Price, Numbed
Restaurants	Name, Latitude, Longitude, Price, Cuisinetype, Opentime, Endtime, Recommendedfood
Transportation	Transportation in specific city including walk, metro and taxi
IntercityTransport	Flight: FlightID, From, To, BeginTime, EndTime, Duration, Cost Train: TrainID, TrainType, From, To, BeginTime, EndTime, Duration, Cost
Poi	Names of POIs(including intercity transportation hub) and their coordinates

Table 4: Sandbox Information

concepts from sandbox information (e.g., restaurant_type(act, city) == "Beijing Cuisine").

Integration with Neuro-Symbolic Agents. (1) NL2DSL Translation: Convert user queries into DSL using LLMs (e.g., "Try local food" → restaurant_type(POI, city) == "Beijing Cuisine" when the destination city is Beijing). (2) Symbolic Validation: Execute DSL code to verify plans against logical constraints. (3) Search Optimization: Use DSL-defined preferences (e.g., minimize(transport_time)) to rank candidate itineraries.

B.3 Query Synthesis

We designed common travel information (origin, destination, days, number of people) and logical constraints based on the nature of travel tasks. To facilitate scalable queries for ChinaTravel, we randomly constructed query skeletons from the aforementioned information and used advanced LLMs to generate natural language queries from these skeletons. In practice, we provide the LLMs with more intuitive hard logic constraints to ensure the LLMs do not make mistakes and use a Python script to convert it after generating the query. The automatically generated data is categorized into two difficulty levels: In the *Easy* level, user inputs encompass a single logical requirement, sourced from categories such as transportation, restaurants, attractions, and accommodations. In the *Medium* level, user inputs involve 2 to 5 logical requirements, introducing more complex constraints. During the generation, we encourage the LLMs to provide varied and human-like expressions, necessitating a deeper understanding and processing to accurately interpret and fulfill the user's needs. For instance, the logical requirement "taste Beijing cuisine" could correspond to the natural language query: "Try local food in Beijing." We utilize prompt engineering to guide LLMs in refining natural language expressions to facilitate automated generation. One of the prompts is shown in Figure 8. Several examples of generated data is in Figure 9.

As a result, we obtain the synthetic queries across diverse travel requirements, including 28 restaurant types, 23 attraction types, 29 hotel features, and more than 130 specific POIs.

B.4 Data Diversity and Bias Mitigation

This subsection provides a detailed analysis of ChinaTravel's hybrid query design, addressing concerns about synthetic data limitations and real-world representativeness.

ChinaTravel integrates both synthetic and human-authored queries to balance scalability and realism. When synthesizing data, we randomly constructed constraints based on the types and specific visit requirements of POIs such as restaurants, accommodations, transports, and attractions, thereby ensuring the diversity of the dataset. The human query subset comprises 154 samples collected through structured questionnaires, which introduce complex real-world constraints such as time-bound returns (e.g., explicit requirements like "return before 7 PM") and activity-specific budget allocations. These queries also incorporate colloquial expressions that reflect native Chinese travel preferences, exemplified by phrases like local specialty foods frequented by residents. The synthetic queries are generated through LLM-based paraphrasing techniques and systematically categorized into two tiers:

Tool	API	Docs
Attractions	attractions_keys(city)	Return a list of (key, type) pairs of the attractions data.
	attractions_select(city, key, func)	Return a DataFrame with data filtered by the specified key with the specified function.
	attractions_id_is_open(city, id, time)	Return whether the attraction with the specified ID is open at the specified time.
	attractions_nearby(city, point, topk, dist)	Return the top K attractions within the specified distance of the location.
	attractions_types	Return a list of unique attraction types.
Accommodations	accommodations_keys(city)	Return a list of (key, type) pairs of the accommodations data.
	accommodations_select(city, key, func)	Return a DataFrame with data filtered by the specified key with the specified function.
	accommodations_nearby(city, point, topk, dist)	Return the top K accommodations within the specified distance of the location.
Restaurants	restaurants_keys(city)	Return a list of (key, type) pairs of the restaurants data.
	restaurants_select(city, key, func)	Return a DataFrame with data filtered by the specified key with the specified function.
	restaurants_id_is_open(city, id, time)	Return whether the restaurant with the specified ID is open at the specified time.
	restaurants_nearby(city, point, topk, dist)	Return the top K restaurants within the specified distance of the location.
	restaurants_with_recommended_food(city, food)	Return all restaurants with the specified food in their recommended dishes.
	restaurants_cuisine(city)	Return a list of unique restaurant cuisines.
Transportation	goto(city, start, end, start_time, transport_type)	Return a list of transportation options between two locations with the specified departure time and transportation mode.
IntercityTransport	intercity_transport_select(start_city, end_city, intercity_type, earliest_leave_time)	Return the intercity transportation information between two cities.
Others	notedown(description, content)	Write the specified content to the notebook
	plan(query)	Generates a plan based on the notebook content and query and report the plan is done.
	next_page()	Get the next page of the latest Result history if it exists. Because of the length limited, all returned DataFrame information is split into 10 rows per page.

Table 5: APIs

Query in Chinese (from easy subset):当前位置成都。我和朋友两个人想去南京玩 2 天，住一间双床房，酒店要可以打牌，请给我一个旅行规划。 Current location: Chengdu. My friend and I want to go to Nanjing for 2 days. We need a twin room in a hotel where we can play cards. Please provide a travel plan for us.
accommodation_type_set=set() for activity in allactivities(plan):   if activity_type(activity) == 'accommodation': accommodation_type_set.add(accommodation_type(activity, target_city(plan))) result=({'棋牌室'}<=accommodation_type_set)
Query in Chinese (from medium subset): 当前位置成都。我一个人想去重庆玩 2 天，预算 3000 人民币，坐火车往返，想吃火锅，想去洪崖洞。 Current location: Chengdu. I want to travel alone to Chongqing for 2 days with a budget of 3000 RMB. I plan to take the train, want to eat hotpot, and visit Hongya Cave.

total_cost=0
for activity in allactivities(plan):
    total_cost+=activity_cost(activity)
    total_cost += innercity_transport_cost(activity_transports(activity))
result=(total_cost<=3000)
intercity_transport_set=set()
for activity in allactivities(plan):
    if activity_type(activity) in ['train', 'airplane']: intercity_transport_set.add(intercity_transport_type(activity))
result=({'train'}==intercity_transport_set)"
restaurant_type_set=set()
for activity in allactivities(plan):
    if activity_type(activity) in ['breakfast', 'lunch', 'dinner']: restaurant_type_set.add(restaurant_type(activity, target_city(plan)))
result=({'火锅'}<=restaurant_type_set)
attraction_name_set=set()\nfor activity in allactivities(plan):
    if activity_type(activity)=='attraction': attraction_name_set.add(activity_position(activity))
result=({'洪崖洞'}<=attraction_name_set)

Query in Chinese (from human subset): [当前位置南京,目标位置武汉,旅行人数 2,旅行天数 3] 我们 2 人想去武汉玩 3 天，主要想体验武汉的一些有些历史的区域，同时还想尝一尝本地人常去吃的特色美食，怎么规划行程。 English translation: [Current location: Nanjing, Destination: Wuhan, Number of travelers: 2, Travel days: 3] The two of us want to visit Wuhan for 3 days. We mainly want to experience some of the historical areas in Wuhan and also try the local specialty foods that residents often eat. How should we plan our itinerary?

attraction_type_set=set()
    for activity in allactivities(plan):
        if activity_type(activity)=='attraction': attraction_type_set.add(attraction_type(activity, target_city(plan)))
result=({'历史古迹'}<=attraction_type_set)"
restaurant_type_set=set()\nfor activity in allactivities(plan):
    if activity_type(activity) in ['breakfast', 'lunch', 'dinner']: restaurant_type_set.add(restaurant_type(activity, target_city(plan)))
result=({'湖北菜'}<=restaurant_type_set)"

Query in Chinese (from human subset): [当前位置南京,目标位置杭州,旅行人数 2,旅行天数 3] 我们打算去杭州看西湖，预算 2000，给我一个旅游安排。 [Current location: Nanjing, Destination: Hangzhou, Number of travelers: 2, Number of travel days: 3] We plan to visit West Lake in Hangzhou with a budget of 2000. Please provide me with a travel itinerary.

attraction_name_set=set()
for activity in allactivities(plan):
    if activity_type(activity)=='attraction': attraction_name_set.add(activity_position(activity))
result=({'西湖风景名胜区'}<=attraction_name_set)
total_cost=0
for activity in allactivities(plan):
    total_cost+=activity_cost(activity)
        total_cost += innercity_transport_cost(activity_transports(activity))
result=(total_cost<=2000)"

Figure 7: Examples of travel requirements and their DSL expressions.

An Example of Prompts for Data Generation

你是一个用户，你想请ai制定一个旅行规划，请根据以下的例子构建一些自然语言的询问，并提供对应的逻辑约束表达。注意tickets和people_number一样。
例子：
JSON：
```
{
    "start_city": "北京",
    "target_city": "南京",
    "hard_logic": [
        "days==2",
        "people_number==1",
        "tickets==1",
        "{'南京大排档'} <= restaurant_names",
    ],
    "nature_language": "当前位置北京。我一个人想去南京玩2天，想吃南京大排档，请给我一个旅行规划。"
}
```
使用如下的餐饮。
店名：{}
即要求restaurant_names包含这个店。
注意，餐饮不一定完全按照提供的特征的名字来，可以使用近义词，比如如果提供的是泳池，可以使用想在酒店游泳这样的自然语言询问
注意，你现在的出发地点为{},目标地点为{}。人数{},天数{}
现在请给一个json询问,
JSON：

```
# You are a user who wants to ask an AI agent to help you plan a
    trip. Please construct some natural language inquiries based
    on the following example and provide the corresponding
   logical constraint expressions. Note that "tickets" and "
   people_number" are the same.
# Example:
# JSON:
# {}
# Use the following restaurants.
# Restaurant name: {}
# This means that "restaurant_names" should include this
   restaurant.
# The dining options may not always be exactly as described by
   the provided features; synonyms can be used. For example, if
   the hotel's feature is a pool, you could ask naturally in
   language like "I want to swim in the hotel pool."
# Now, your departure location is {}, and your destination is
   {}. The number of people is {}, and the number of days is {}.
# Now please provide a JSON inquiry.
# JSON:
```

Figure 8: An example of prompts for data generation. This example is about restaurant_name. By replacing this with other constraints or combining multiple constraints, we can generate data with different levels of difficulty based on different constraints.

Examples of Generated Data

Example 1

```
{
    "start_city": "杭州",
    "target_city": "上海",
    "hard_logic": [
        "days==2",
        "people_number==1",
        "tickets==1",
        "{'本帮菜'} <= food_type"
    ],
    "nature_language": "当前位置杭州。我一个人想去上海玩2天，想尝试当地的特色菜，请给我一个旅行规划。"
}
```

Example 2

```
{
    "start_city": "深圳",
    "target_city": "北京",
    "hard_logic": [
        "days==2",
        "people_number==3",
        "intercity_transport=={'airplane'}",
        "tickets==3",
        "rooms==3",
        "room_type==1"
    ],
    "nature_language": "当前位置深圳。我们三个人计划去北京玩两天，选择飞机出行，开三间大床房。请给我一个旅行规划。"
}
```

Example 3

```
{
    "start_city": "重庆",
    "target_city": "苏州",
    "hard_logic": [
        "days==3",
        "people_number==3",
        "cost<=7300",
        "{'日本料理'} <= food_type",
        "intercity_transport=={'train'}",
         "tickets==3",
        "rooms==2",
        "room_type==2"
    ],
    "nature_language": "当前位置重庆。我们三个人计划去苏州玩三天，选择火车出行，想吃日本料理，预算7300元，开两间双床房。请给我一个旅行规划。"
}
```

Figure 9: Examples of Generated Data

Logical Constraint	
Transportation	The required type of transportation.
Attraction	The required type or specified attractions.
Restruants	The required type or specified restruants.
Accommodation	The number of rooms and the room type must meet the requirements.
	The required features or specified hotels.
Budget	The total cost is within required budget.
Unseen Logical Constraints in Human data	
POIs	The negation/conjunction/disjunction of given POIs
Time	The duration of specific activities is within the limitation
Budget	The cost of specific activities is within the required budget

Table 6: Descriptions of **Logical Constraints** for two benchmarks. Constraints in black are common in both TravelPlanner and ChinaTravel. Metrics in brown are the metrics only in our benchmark.

Preference Requirements	
Daily attractions ↑	Visit as many attractions as possible
Transport time ↓	Minimize the travel time between POIs
Transport time to the restaurants ↓	Minimize the travel time to restaurants
Food cost ratio ↑	Maximize the proportion of dining expenses
Hotel cost ↓	Minimize accommodation costs
Distance to POI ↓	Visit places as close to {POI} as possible

Table 7: Descriptions of **Preference Requirements** in ChinaTravel benchmark.

Easy-tier queries contain single logical constraints (e.g., specific cuisine requirements), while Medium-tier queries combine 3–5 interdependent constraints (e.g., compound conditions like "budget $\leq$ 3000 CNY + train transport + hotpot dining").

To mitigate synthetic data bias and enhance diversity, three primary strategies were implemented. First, constraint combinations were deliberately diversified across temporal, spatial, and cost dimensions, as detailed in Table 6. Second, a human validation layer filters out unrealistic LLM-generated queries, such as physically implausible itineraries like "visiting 10 attractions within one day." Third, the DSL framework enables compositional generalization of requirements, supporting open-ended constraint combinations through its formal syntax shown in Table 2.

The current human query subset remains limited by annotation costs, as discussed in the limitation section. In future work, we will advance data collection by integrating LLMs with real user queries to automate and diversify the generation of human-like queries. Additionally, all human queries and automated synthesis tools will be publicly released to support community-driven benchmark extensions.

B.5 Data with Preference

We introduce six common preferences from user surveys to construct the preference sub-datasets. Table 12 provides a summary of these preferences.

The corresponding DSL could be formulated as follows.

```
# The number of attractions visited
count = 0
for act_i in all_activities(plan):
  if activity_type(act_i)=="attraction":
     count = count + 1
return count
```

```
# The average travel time between POIs
time_cost = 0
transport_count = 0
for activity in allactivities(plan):
        transports =
    activity_transports(activity)
    transport_count += 1
        time_cost +=
        innercity_transport_time(
        transports)
average_time_cost = time_cost /
    transport_count if transport_count >
     0 else -1
return average_time_cost
```

Function Name	Meaning	Implementation
day_count	total days in the plan	`def day_count(plan):` `    return len(plan["itinerary"])`
people_count	number of people in the trip	`def people_count(plan):` `    return plan["people_number"]`
start_city	start city of the plan	`def start_city(plan):` `    return plan["start_city"]`
target_city	target city of the plan	`def target_city(plan):` `    return plan["target_city"]`

allactivities	all the activities in the plan	`def allactivities(plan):`
		`    activity_list = []`
		`    for day_activity in plan["itinerary"]:`
		`        for act in day_activity["activities"]:`
		`            activity_list.append(act)`
		`    return activity_list`

allactivities_-count	the number of activities in the plan	`def allactivities_count(plan):`
		`    count = 0`
		`    for day_activity in plan["itinerary"]:`
		`        count += \`
		`            len(day_activity["activities"])`
		`    return count`

dayactivities	all the activities in the specific day [1, 2, 3, ...]	`def dayactivities(plan, day):`
		`    activity_list = []`
		`    for act in plan["itinerary"]\`
		`        [day - 1]["activities"]:`
		`        activity_list.append(act)`
		`    return activity_list`

activity_cost	the cost of specific activity without transport cost	`def activity_cost(activity):` `    return activity.get("cost", 0)`
activity_posi-tion	the position name of specific activity	`def activity_position(activity):` `    return activity.get("position", "")`
activity_price	the price of specific activity	`def activity_price(activity):` `    return activity.get("price", 0)`
activity_type	the type of specific activity	`def activity_type(activity):` `    return activity.get("type", "")`
activity_tickets	the number of tickets needed for specific activity	`def activity_tickets(activity):` `    return activity.get("tickets", 0)`
activity_trans-ports	the transport information of specific activity	`def activity_transports(activity):` `    return activity.get("transports", [])`
activity_-start_time	the start time of specific activ-ity	`def activity_start_time(activity):` `    return activity.get("start_time")`
activity_-end_time	the end time of specific activ-ity	`def activity_end_time(activity):` `    return activity.get("end_time")`

Table 8: Concept Function

<table>
<tr><th>Function Name</th><th>Meaning</th><th>Implementation</th></tr>
<tr><td>activity_time</td><td>the duration of specific activity</td><td>

```python
def activity_time(activity):
    start_time = activity.get("start_time")
    end_time = activity.get("end_time")
    if start_time and end_time:
        st_h, st_m = \
            map(int, start_time.split(":"))
        ed_h, ed_m = \
            map(int, end_time.split(":"))
        return \
            (ed_m - st_m) + (ed_h - st_h) * 60
    return -1
```

</td></tr>
<tr><td>poi_recommend_time</td><td>the recommend time of specific poi(attraction) in the city</td><td>

```python
def poi_recommend_time(city, poi):
    select = Attractions().select
    attrction_info = \
        select(city, key="name",
               func=lambda x: x == poi).iloc[0]
    recommend_time = \
        (attrction_info["recommendmintime"]) \
        * 60
    return recommend_time
```

</td></tr>
<tr><td>poi_distance</td><td>the distance between two POIs in the city</td><td>

```python
def poi_distance(city, poi1, poi2):
    start_time="00:00"
    transport_type="walk"
    goto = Transportation().goto
    return goto(city, poi1, poi2, start_time,
                transport_type)[0]["distance"]
```

</td></tr>
<tr><td>innercity_transport_cost</td><td>the total cost of specific innercity transport</td><td>

```python
def innercity_transport_cost(transports, mode):
    cost = 0
    for transport in transports:
        if node is None or \
            transport.get("type") == node:
            cost += transport.get("cost", 0)
    return cost
```

</td></tr>
<tr><td>innercity_transport_price</td><td>the price of innercity transport</td><td>

```python
def innercity_transport_price(transports):
    price = 0
    for transport in transports:
        price += transport["price"]
    return price
```

</td></tr>
<tr><td>innercity_transport_distance</td><td>the distance of innercity transport</td><td>

```python
def innercity_transport_distance\
    (transports, mode=None):
    distance = 0
    for transport in transports:
        if mode is None or \
            transport.get("type") == mode:
            distance += \
                transport.get("distance", 0)
    return distance
```

</td></tr>
<tr><td>innercity_transport_time</td><td>the duration of innercity transport</td><td>

```python
def innercity_transport_time(transports):
    def calc_time_delta(end_time, start_time):
        hour1, minu1 = \
            int(end_time.split(":")[0]), \
                int(end_time.split(":")[1])
        hour2, minu2 = \
            int(start_time.split(":")[0]), \
                int(start_time.split(":")[1])
        return (hour1 - hour2) * 60\
            + (minu1 - minu2)
```

</td></tr>
</table>

Table 9: Concept Function

<table>
<thead>
<tr><th>Function Name</th><th>Meaning</th><th>Implementation</th></tr>
</thead>
<tbody>
<tr><td>metro_tickets</td><td>the number of metro tickets if the type of transport is metro</td><td><pre><code>def metro_tickets(transports):
    return transports[1]["tickets"]</code></pre></td></tr>
<tr><td>taxi_cars</td><td>the number of taxi cars if the type of transport is taxi</td><td><pre><code>def taxi_cars(transports):
    return transports[0]["cars"]</code></pre></td></tr>
<tr><td>room_count</td><td>the number of rooms of accommodation</td><td><pre><code>def room_count(activity):
    return activity.get("rooms", 0)</code></pre></td></tr>
<tr><td>room_count</td><td>the number of rooms of accommodation</td><td><pre><code>def room_count(activity):
    return activity.get("rooms", 0)</code></pre></td></tr>
<tr><td>room_type</td><td>the type of room of accommodation</td><td><pre><code>def room_type(activity):
    return activity.get("room_type", 0)</code></pre></td></tr>
<tr><td>restaurant_type</td><td>the type of restaurant's cuisine in the target city</td><td><pre><code>def restaurant_type(activity, target_city):
    restaurants = Restaurants()
    select_food_type = \
        restaurants.select(
        target_city, key="name",
        func=lambda x: x == activity["position"]
)["cuisine"]
    if not select_food_type.empty:
        return select_food_type.iloc[0]
    return ""</code></pre></td></tr>
<tr><td>attraction_type</td><td>the type of attraction in the target city</td><td><pre><code>def attraction_type(activity, target_city):
    attractions = Attractions()
    select_attr_type = \
        attractions.select(
        target_city, key="name",
        func=lambda x: x == activity["position"]
)["type"]
    if not select_attr_type.empty:
        return select_attr_type.iloc[0]
    return ""</code></pre></td></tr>
<tr><td>accommodation_type</td><td>the feature of accommodation in the target city</td><td><pre><code>def accommodation_type(activity, target_city):
    accommodations = Accommodations()
    select_hotel_type = \
        accommodations.select(
        target_city, key="name",
        func=lambda x: x == activity["position"]
)["featurehoteltype"]
    if not select_hotel_type.empty:
        return select_hotel_type.iloc[0]
    return ""</code></pre></td></tr>
<tr><td>innercity_transport_type</td><td>the type of innercity transport</td><td><pre><code>def innercity_transport_type(transports):
    if len(transports) == 3:
        return transports[1]["mode"]
    elif len(transports) == 1:
        return transports[0]["mode"]
    return ""</code></pre></td></tr>
<tr><td>intercity_transport_type</td><td>the type of intercity transport</td><td><pre><code>def intercity_transport_type(activity):
    return activity.get("type", "")</code></pre></td></tr>
</tbody>
</table>

Table 10: Concept Function

Function Name	Meaning	Implementation
innercity_-transport_-start_time	the start time of innercity transport	`def innercity_transport_start_time(transports):` `    return transports[0]["start_time"]`
innercity_-transport_-end_time	the end time of innercity transport	`def intercity_transport_end_time(transports):` `    return transports[-1]["end_time"]`

intercity_-transport_-origin	the origin city of intercity transport	`def intercity_transport_origin(activity):`
		`    if "start" in activity:`
		`        for city in city_list:`
		`            if city in activity["start"]:`
		`                return city`
		`    return ""`

intercity_-transport_-destination	tthe destination city of intercity transport	`def intercity_transport_destination(activity):`
		`    if "end" in activity:`
		`        for city in city_list:`
		`            if city in activity["end"]:`
		`                return city`
		`    return ""`

Table 11: Concept Function

```
# The average travel time to restaurants
restaurant_count = 0
time_cost = 0
for activity in allactivities(plan):
    if activity_type(activity) in ['
        breakfast', 'lunch', 'dinner']:
        restaurant_count += 1
        time_cost +=
            innercity_transport_time(
            activity_transports(activity
            ))
if restaurant_count == 0:
    average_time_cost = -1
else:
    average_time_cost = time_cost /
        restaurant_count
return average_time_cost
```

```
# The ratio of food cost
food_cost = 0
total_cost = 0
for activity in allactivities(plan):
    total_cost += activity_cost(activity
        )
    total_cost +=
        innercity_transport_cost(
        activity_transports(activity))
    if activity_type(activity) in ['
        breakfast', 'lunch', 'dinner']:
        food_cost += activity_cost(
            activity)
food_cost_ratio = food_cost / total_cost
     if total_cost > 0 else -1
return food_cost_ratio
```

```
# The cost of accommodations
accommodation_cost = 0
for activity in allactivities(plan):
    if activity_type(activity) == '
        accommodation':
        accommodation_cost +=
            activity_cost(activity)"
return accommodation_cost
```

```
# The average distance to poi (e.g. xxx)
target_poi = 'xxx'
poi_list = list()
total_distance = 0
poi_count=0
city = target_city(plan)
for activity in allactivities(plan):
    if activity_type(activity) in ['
        breakfast', 'lunch', 'dinner', '
        accommodation', 'attraction']:
        poi_list.append(
            activity_position(activity))
for poi in poi_list:
    total_distance += poi_distance(city,
         target_poi, poi)
    poi_count += 1
average_dist_cost = total_distance /
    poi_count if poi_count > 0 else -1
return average_dist_cost
```

B.6 Benchmark Difficulty and Applicability

While the Human subset presents significant challenges, the baseline NeSy solution has achieved 60.6% and 46.7% FPR on Easy and Medium subsets, respectively, providing developers with actionable validation points for initial testing and refinement. Additionally, the Human subset's extreme difficulty arises from open language reasoning and unseen concept composition, key challenges absent in prior benchmarks but unavoidable in practice. By explicitly formalizing these challenges, ChinaTravel has provided a roadmap for advancing agents toward real-world robustness. Despite current LLMs' limitations in handling unseen combinations, their success in code generation suggests that post-training with DSL may enhance their understanding of diverse travel needs, moving toward real-world applications.

C Discussion with TravelPlanner

TravelPlanner's logical constraints contain the total cost, 15 cuisines, 5 house rules, 3 room types, and 3 intercity transports. ChinaTravel's logical constraints contain the total cost, 42 cuisines, 15 attraction types, 78 hotel features, 2 room types, 2 intercity-transports types, 3 inner-city-transports types, and specific POI names (attractions, restaurants, hotels). Crucially, our benchmark introduces compositional constraints derived from human queries (e.g., "return before 7 PM", "cost of intercity transports"), reflecting real-world complexity. The key advancement lies in addressing open-language reasoning and unseen concept composition, which represent significant challenges beyond TravelPlanner's scope. Our Domain-Specific Language (DSL) enables automated validation of these combinatorial requirements, bridging the gap between synthetic and real-world needs.

We also provide some example queries and corresponding examples from the TravelPlanner at each level in Figure 15, 16, and 17.

As shown in Figure 15, in the "easy level", TravelPlanner only includes constraints on cost. In contrast, ChinaTravel demonstrates significant advantages over TravelPlanner, particularly in terms of personalized support for specific Points of Interest (POIs) and more realistic transportation and time management. These advantages are crucial for developing and evaluating language agents that can handle real-world travel planning scenarios effectively. ChinaTravel allows users to directly specify POI names, such as "Nanjing DaPaXiang" or "HuQiu Mountain Scenic Area," requiring the agent to precisely match the entity information from the travel sandbox.

As shown in Figure 16, in the medium set, TravelPlanner includes queries with two types of constraints: cost and cuisine, or cost and accommodation. In contrast, ChinaTravel includes queries with 2 to 5 types of constraints, reflecting more complex and diverse multi-constraint requirements. This difference highlights the ability of ChinaTravel to handle more realistic and varied travel planning scenarios.

As shown in Figure 17, TravelPlanner includes queries with multiple constraints, such as cost, accommodation type, and cuisine preferences. However, ChinaTravel goes a step further by including queries with unseen logical constraints and more colloquial expressions. These unseen logical constraints and colloquial expressions are essential for travel planning agents to handle real-world users effectively. They reflect the complexity and diversity of real-world travel planning scenarios, where users may have diverse requirements that need to be understood and addressed. By incorporating these elements, ChinaTravel bridges the gap between current academic research benchmarks and real-world application demands, making it a more comprehensive and realistic benchmark for evaluating the capabilities of travel planning agents.

D NeSy Planning

Since the Z3 solver from (Hao et al., 2024) would restructure the tool API to return travel information expressed in specific Z3 variables, which may not be feasible given that APIs in the real world are typically black boxes that agents can only call. Following their two-stage solution, we first extract logical constraints from natural language. Based on these constraints, we implement a step-by-step plan generation process using depth-first search, mimicking how humans plan to travel by arranging activities one by one. As shown in Fig. 3, we first translate the natural languages to logical constraints through prompting. generate the next activity type based on the current plan, and then recursively generate the next activity until the goal is reached. The generated plan is then used to solve the problem. In the second step, we define the rule-based activity selection and score function. For example, if the current time is in the [10:30, 12:30] and there is no scheduled lunch in the current plan, then the agent should find a restaurant to have lunch at this time. If the current time is after 22:00 and there are no open-time attractions nearby, the agent should choose to return to the hotel. For the score function, we select the restaurants that satisfy the required cuisine and sort the candidates by the price if there a budget constraints in the constraints C. These ranking functions will help us to find a feasible solution as soon as possible. In ChinaTravel, the duration arrangement of activities is continuous and difficult to enumerate and search. We pre-define a meal or a visit to an attraction as 90 minutes, and when there are less than 90 minutes until closing time, the event continues until the closing time. Given these designs, we adapt the neural-symbolic solution into a multi-POI planning problem and evaluate it in the ChinaTravel benchmark.

Given that some queries are particularly challenging due to the limited number of feasible plans, we set the maximum runtime for the symbolic sketch from interactive search to 5 minutes per query, excluding the LLM inference time, to ensure a fair comparison across different models. If a plan satisfying the generated DSL validation is found within the time limit, it is returned directly. Otherwise, the program halts when the time limit is reached, and the plan that satisfies environmental constraints while achieving the highest number of validation code successes among all intermediate results is returned. In cases where no environment-compliant plan is identified, the partially completed plan generated up to that point is returned.

In the Figure 18, 19 and 20, we provide the prompts of the LLM POI-ranking phases.

E Evaluation Metric in Competition

The Delivery Rate (DR), Environmental Pass Rate (EPR), Logical Pass Rate (LPR), and Final Pass Rate (FPR) have been detailed in TravelPlanner (Xie et al., 2024). To make the paper more self-contained, we also provide the corresponding definition here.

Delivery Rate: This metric assesses whether agents can successfully deliver a formatted plan. For the Nesy planning, if a solution that satisfies the logical constraints has not been found by the time out, the search is terminated, and the current solution that satisfies the environmental constraints is returned. If no solution that satisfies the environmental constraints is obtained, an empty plan is returned. Therefore, unlike the pure LLM method, which primarily assesses the Delivery Rate based on whether the output meets the formatting requirements, the nesy planning method, which uses depth-first-search to arrange POIs one by one, shows differences in the Delivery Rate. These differences mainly reflect the proportion of effective solutions obtained within a limited time based on the LLM's POI recommendation. This proportion demonstrates the degree to which the LLM prioritizes POI arrangements from a natural language perspective and meets formalized logical requirements. The more accurately the LLM can arrange POIs that are beneficial for long-horizon planning, the more likely it is to obtain effective solutions and improve the Delivery Rate.

Environmental Pass Rate Comprising the environmental dimensions (as detailed in Tab. 1), this metric evaluates whether a language agent can accurately incorporate sandbox information into their generated plans.

$$EPR-micro = \frac{\sum_{p\in P}\sum_{c\in Env}\mathbb{1}_{passed(c,p)}}{|P|*|Env|}$$

$$EPR-macro = \frac{\sum_{p\in P}\prod_{c\in Env}\mathbb{1}_{passed(c,p)}}{|P|}$$

Logical Pass Rate Comprising the logical dimensions (as detailed in Tab. 6), this metric evaluates whether a language agent can accurately meet the personalized requirements of the queries.

$$LPR-micro = \frac{\sum_{p\in P}\sum_{c\in C_p}\mathbb{1}_{passed(C_p,p)}}{\sum_{p\in P}|C_p|}$$

$$LPR-macro = \frac{\sum_{p\in P}\prod_{c\in C_p}\mathbb{1}_{passed(C_p,p)}}{|P|}$$

Final Pass Rate This metric represents the proportion of feasible plans that meet all aforementioned constraints among all tested plans. It serves as an indicator of agents' proficiency in producing plans that meet a practical standard.

$$FPR = \frac{\sum_{p\in P}\mathbb{1}_{passed(Env,p)}\cdot\mathbb{1}_{passed(C_p,p)}}{|P|}$$

Preference Ranking To systematically evaluate the satisfaction of soft user preferences in travel planning, we introduce a Preference Ranking metric that quantifies the alignment of generated itineraries with diverse user requirements. Each preference (e.g., "maximize attraction visits" or "minimize transportation time") is formalized into a Domain-Specific Language (DSL)-based concept, enabling automated numerical extraction from plans. For

Algorithm 1 Depth-First Greedy Search

```
Require: Constraints C, current plan p,
  if the least activity is an intercity-transport from destination to origin then
    return ConstraintValidation(p, C), p        ▹ The plan p is finished, return the validation result.
  end if
  type = GetNextActivityType(p)                 ▹ Select the next type of activities, e.g. lunch, attraction.
  candidates = ToolUse(type)                    ▹ Collect the corresponding information for the activity type
  scores = LLMScore(candidates, p, C)           ▹ Score candidates through constraints C.
  for activity in candidates do
    p.push(activity)                            ▹ Perform a greedy search with priority ranking.
    flag, p = Depth-FirstGreedySearch(C, p)
    if flag then
      return True, p                            ▹ Return the solution p if the validation is passed.
    end if
    p.pop(activity)
  end for
  return False, p                               ▹ Fail to find a solution with the given conditions.
```

instance, the preference for "visiting more attractions" is translated into a DSL function that counts the total attraction-type activities in a plan, while "minimizing dining costs" is operationalized via cumulative expense calculations for meal-related activities.

The Preference Ranking metric operates in two steps: 1) Quantification: Execute DSL code to compute concept-specific scores (e.g., attraction count, transport time) for each generated plan. 2) Ranking: Compare methods (e.g., BQ vs. PEQ vs. PDS) by ranking their concept values, prioritizing higher values for maximization goals (↑) and lower values for minimization goals (↓). 3) Aggregation: Calculate the average ranking on the given samples.

F Additional Experimental Results

F.1 Multi-Preference Comparison

For multi-preference scenarios (e.g., balancing "attraction visits ↑" and "transport time ↓"), we adopt an averaged aggregation approach, where rankings reflect the combined performance across all preferences. This framework ensures scalability and objectivity.

To rigorously evaluate the ability of language agents to balance multiple soft constraints, we constructed 15 test subsets by pairing six user preferences (P0–P5) into all possible combinations (e.g., "P0 + P1"). Each subset contains queries with two preference requirements. We compared two methods, Baseline Query (BQ) and Preference-Enhanced Query (PEQ), by quantifying their performance through our DSL-based Preference Ranking metric. For each subset, we extracted numerical scores for both preferences (Value-1 and Value-2), computed individual rankings (Rank-1, Rank-2), and derived an aggregated ranking (Agg. Rank.) to reflect overall performance. The results are provided in the Table 12.

From these results, we could find that: (1) **PEQ Outperforms BQ in Most Scenarios**: In 10/15 combinations, PEQ achieves superior aggregated rankings (Aggregated Ranking = 1.43 vs. BQ's 1.56). Notably, PEQ demonstrates stable improvements on preferences P3 (e.g., maximizing dining quality↑) and P4 (e.g., minimizing accommodation costs↓). For instance: In "P0↑ + P4↓", PEQ reduces accommodation costs by 64% (Value-2: 441 vs. BQ's 1221) while maintaining high attraction counts (Value-1: 0.97 vs. 0.79). For "P3↑ + P4↓", PEQ simultaneously improves dining quality (Value-1: 0.26 vs. BQ's 0.18) and lowers costs (Value-2: 531 vs. 1229). This stability likely stems from the direct impact of POI selection on these preferences. LLMs in PEQ effectively prioritize low-cost hotels or high-quality restaurants through natural language hints (e.g., "reduce the cost on accommodations"), enabling explicit alignment with P3 and P4 requirements. (2) **Challenges in Balancing Multiple Preferences**: The results also reveal inherent difficulties in harmonizing conflicting preferences, particularly when optimizing one requirement necessitates sacrificing another. Notably, in the P0↑ + P1↓ scenario, PEQ underperforms BQ on both preferences, highlighting the

Preference Combination	Vaule-1		Vaule-2		Rank-1		Rank-2		Agg. Rank.	
	BQ	PEQ	BQ	PEQ	BQ	PEQ	BQ	PEQ	BQ	PEQ
P0 $\uparrow$, P1 $\downarrow$	0.79	**0.83**	**28.0**	29.7	**1.44**	1.55	**1.44**	1.55	**1.44**	1.55
P0 $\uparrow$, P2 $\downarrow$	0.82	**1.26**	**29.0**	31.9	1.56	**1.43**	**1.43**	1.56	1.5	1.5
P0 $\uparrow$, P3 $\uparrow$	0.81	**0.94**	0.18	**0.20**	**1.42**	1.57	1.59	**1.40**	1.51	**1.48**
P0 $\uparrow$, P4 $\downarrow$	0.79	**0.97**	1221	**441**	1.46	**1.53**	1.73	**1.26**	1.59	**1.40**
P0 $\uparrow$, P5 $\downarrow$	0.78	**0.91**	**33.6**	34.0	**1.37**	1.62	1.70	**1.29**	1.54	**1.45**
P1 $\downarrow$, P2 $\downarrow$	28.2	**27.8**	**26.6**	30.1	1.62	**1.37**	**1.48**	1.51	1.55	**1.44**
P1 $\downarrow$, P3 $\uparrow$	**28.2**	36.2	0.20	**0.27**	**1.31**	1.68	1.6	**1.4**	**1.45**	1.54
P1 $\downarrow$, P4 $\downarrow$	**30.3**	44.8	1440	**585**	**1.14**	1.85	1.77	**1.22**	**1.45**	1.54
P1 $\downarrow$, P5 $\downarrow$	**30.1**	38.3	30.7	**30.2**	**1.27**	1.72	1.69	**1.30**	**1.48**	1.51
P2 $\downarrow$, P3 $\uparrow$	24.7	**23.3**	0.27	0.27	**1.43**	1.56	1.60	**1.39**	1.52	**1.47**
P2 $\downarrow$, P4 $\downarrow$	24.1	**21.1**	1687	**719**	1.51	**1.48**	1.89	**1.10**	1.70	**1.29**
P2 $\downarrow$, P5 $\downarrow$	**28.0**	30.8	29.4	**26.0**	1.51	**1.48**	1.89	**1.10**	1.70	**1.29**
P3 $\uparrow$, P4 $\downarrow$	0.18	**0.26**	1229	**531**	1.64	**1.35**	1.69	**1.30**	1.66	**1.33**
P3 $\uparrow$, P5 $\downarrow$	0.22	0.22	33.3	**29.0**	1.51	**1.48**	1.84	**1.15**	1.68	**1.31**
P4 $\downarrow$, P5 $\downarrow$	1366	**767**	33.1	**31.6**	1.67	**1.32**	**1.45**	1.54	1.56	**1.43**
Aggregated Ranking									1.56	**1.43**

Table 12: Multi-Preference Comparison of BQ and PEQ.

inherent difficulty in resolving conflicting objectives. While PEQ marginally improves attraction counts (Value-1: 0.83 vs. BQ's 0.79), it incurs a 5.7% increase in transport time (Value-2: 29.7 vs. BQ's 28.0). This trade-off results in a worse aggregated ranking for PEQ (1.55 vs. BQ's 1.44), indicating that the combined effect of conflicting preferences negates the benefits of natural language guidance. In 9/15 combinations, PEQ improves one preference at the expense of the other. For example: P1↓ + P4↓: PEQ reduces accommodation costs by 59% (Value-2: 585 vs. BQ's 1440) but increases transport time by 48% (Value-1: 44.8 vs. 30.3). The inability to concurrently satisfy both preferences underscores the limitations of current LLM-driven prioritization in handling trade-offs.

Our experiments demonstrate that the neuro-symbolic agent (PEQ), enhanced by LLM-driven POI recommendation, outperforms baseline methods in multi-preference travel planning. By integrating natural language hints to guide POI selection, PEQ effectively translates user requirements into actionable itineraries, demonstrating its capability to handle synergistic preferences. However, balancing inherently conflicting objectives remains challenging. This highlights the need for future advancements, such as domain-specific fine-tuned LLMs to better resolve preference conflicts or multi-objective optimization techniques to systematically navigate trade-offs.

F.2 Open Reasoning with Chinese Context

In this section, we quantitatively compare the reasoning capabilities of LLMs in the context of Chinese travel requirements. Given that many leading LLMs, such as GPT-4, are primarily trained in English corpora, it is essential to evaluate their performance in a Chinese travel planning context to better understand their reasoning abilities. We focus on three LLMs: GPT-4o, DeepSeek-V2.5, and Qwen2.5-7B, which are employed in the main experiments.

Specifically, we analyze the POI matching in the NL2DSL process with varying travel requirements from the synthesized quires and further provide the distribution of the results in Figure 10. The com-

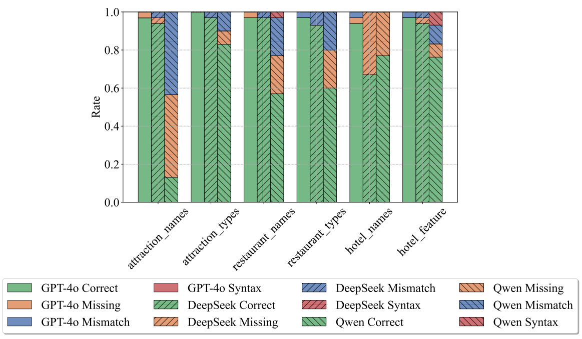


Figure 10: Results Distribution on Synthesized Quires

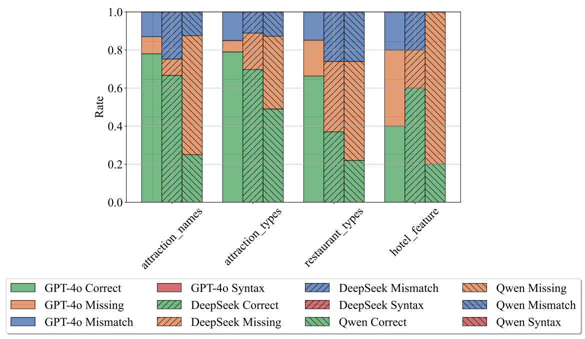


Figure 11: Results Distribution on Human Quires

parative analysis reveals significant disparities in reasoning capabilities across the three LLMs when handling Chinese travel-related queries. DeepSeek-V2.5 demonstrates robust performance in most categories, achieving high accuracy (Correct ≥ 93%) for attraction-names, attraction-types, restaurant-names, and hotel-features. However, its performance sharply declines in hotel-names (Correct: 67%, Missing: 33%), suggesting limited familiarity with Chinese hotel nomenclature or insufficient contextual grounding in this domain. This contrasts with GPT-4o, which excels in hotel-names (Correct: 93%) and achieves perfect accuracy (Correct: 100%) for attraction-types, highlighting its superior cross-lingual transfer capabilities despite being primarily English-trained. Notably, GPT-4o maintains consistent performance across all categories (Correct ≥ 93%), underscoring its balanced reasoning proficiency in Chinese contexts. In stark contrast, Qwen2.5-7B exhibits critical weaknesses, particularly in attraction-names (Correct: 13%, Error: 43%), indicating severe limitations in entity recognition and syntactic coherence for Chinese proper nouns. The pronounced missing rates observed in Qwen2.5-7B (e.g., 43% for attraction-names and 23% for hotel-names) align with its constrained parameter size (7B), which likely impedes its ability to internalize diverse travel requirements or align them with sandbox's POI information.

We further conduct the analysis and provide the results on human queries in Figure 11. The evaluation of human queries reveals critical limitations in LLMs' practical reasoning capabilities that synthetic data fails to expose. DeepSeek-V2.5's accuracy plummets in hotel-feature (Correct: 40% vs. 93% in synthetic data), indicating severe degradation when handling ambiguous or culturally nuanced requirements (e.g., interpreting subjective descriptors like "luxury" or "traditional courtyard-style" in Chinese contexts). GPT-4o similarly exhibits instability, with significant declines in restaurant-types (Correct: 37% vs. 97% in synthetic data) and attractions-type (Correct: 69% vs. 100%), suggesting that its cross-lingual transfer mechanisms falter when confronted with real-world linguistic variability (e.g., colloquial phrasing or dialect influences). This analysis underscores the necessity of introducing human queries into benchmarks when evaluating travel planning, as they reveal critical gaps in open language reasoning for deploying LLMs in real-world travel assistants.

F.3 Analysis of Pure-LLM Methods

Pure LLM-based methods have demonstrated significant shortcomings in constraint satisfaction, as evidenced by their near-zero success rates in benchmarks like TravelPlanner. We also attempt the multi-round refinement methods like Reflexion. While theoretically promising, it is still impractical in our context. In preliminary evaluations, Reflexion not only failed to achieve improvements in constraint satisfaction (consistent 0% FPR) but also incurred prohibitive computational costs due to its reliance on iterative token-heavy interactions. This rendered large-scale evaluation infeasible given our resource constraints. In light of their current limitations in constraint satisfaction, NeSy frameworks remain the effective pathway for real-world travel planning. Therefore, in the main body of this work, we mainly analyze the Nesy method.

In this section, we further summarize the key failure modes of pure-LLM-based methods observed in our experiments:

(1) **Incorrect API Calls:** LLMs frequently generate invalid or hallucinated API calls, leading to cascading errors in downstream planning. For instance, models may query non-existent APIs (e.g., city_transport_select instead of in-

ter_city_transport_select) or misuse parameters (e.g., filtering attractions by an unsupported feature like "bus"). Such errors exhaust API call limits and prevent agents from retrieving essential information.

(2) **Repetitive Output Loops** In iterative planning frameworks like ReAct, LLMs often enter infinite loops when resolving constraints. For example, an agent might repeatedly query transportation details for all candidate attractions, even after selecting one, due to a failure to update its internal state. This behavior mimics the “hallucination loops" reported in TravelPlanner paper.

(3) **Reasoning-Action Inconsistency.** In ReAct framework, the model first reasons and then takes an action. However, the reasoning and the action are not always consistent. For example, the model may reason that the user wants to book a flight, but then take an action to check the information of trains. Another example is that the model may detect that the expenses exceed the budget but does not respond to this and ultimately generates a plan that exceeds the budget.

(4) **Critical Information Missing.** Even when intermediate steps (e.g., API responses) are logged in a "notebook," LLMs frequently omit essential details when synthesizing final plans. A recurring failure is neglecting return transportation (e.g., omitting the train from Shanghai back to Beijing), which violates feasibility constraints.

Figure 12 provides the fail examples of ReAct (one-shot) with DeepSeek, which outperforms other pure-LLM-based methods in the main experiments.

These limitations underscore the inadequacy of pure-LLM-based approaches for deployment in long-horizon and constraint-rich domains like travel planning.

F.4 Holistic Score on Overall Dataset

In Table 13, we provide the holistic scores combining the results on easy, medium, and human subsets to show the overall performance.

G Statements about Scientific Artifacts

The ChinaTravel benchmark is designed to facilitate research in natural language processing and artificial intelligence, specifically for travel planning tasks. ChinaTravel includes a travel sandbox, user queries, and an evaluation framework intended for non-commercial, academic research purposes.

Availability. We will publicly release the ChinaTravel benchmark upon publication to facilitate community research. We look forward to broader adoption and extension of this benchmark.

Licenses. The ChinaTravel benchmark and its associated datasets are licensed under the Creative Commons Attribution-NonCommercial 4.0 International (CC-BY-NC 4.0) license. This license allows for the free use, distribution, and reproduction of the benchmark in any medium, provided that appropriate credit is given to the original authors and the source of the data is acknowledged, and that the use is for non-commercial purposes only.

Data anonymization and offensive content. We anonymized the human queries during collection and instructed participants to avoid including sensitive information. We removed queries containing offensive content during the data cleaning process.

H Statements about Human Participants

We recruited over 250 volunteers through a structured questionnaire to collect authentic Chinese travel requirements. Participants were informed about the public use of their data and instructed to avoid including sensitive personal information. During data cleaning, offensive content and identifiable details were removed. While no explicit ethics board approval is mentioned, we ensured compliance with anonymization practices and obtained participant consent for data inclusion. The final dataset contains 154 human-derived queries reflecting diverse real-world travel needs.

H.1 Instructions Given To Participants

To gather the authentic travel requirements, we collected data through a carefully designed questionnaire. We provided the following instruction information to the participants:

1. The specific constraints the agent can handle and the corresponding details, including the types and specific names of attractions, restaurants, and hotels; requirements for intercity transportation (airplane or train) and urban transportation (walk, taxi or subway); as well as budget limitations for overall expenses or specific activities (such as accommodation and intercity transportation).

2. The necessary information should be provided in the query, including the departure and des-

Incorrect API Calls

Action[20]: goto(city="深圳", start="深圳北站", end="大鹏所城", start_time="19:00", method="bus")
Observation[20]: Invalid method. Only support ['metro', 'walk', 'taxi']
…
Action[25]: goto(city="深圳", start="深圳北站", end="大鹏所城", start_time="19:00", method="bus")
Observation[25]: Invalid method. Only support ['metro', 'walk', 'taxi']
…
Action[30]: goto(city="深圳", start="深圳北站", end="大鹏所城", start_time="19:00", method="bus")
Observation[30]: Invalid method. Only support ['metro', 'walk', 'taxi']

Action[3]: city_transport_select(start_city="成都", end_city="武汉", intercity_type="train", earliest_leave_time="19:00")
Observation[3]: 调用失败，错误信息：name 'city_transport_select' is not defined请思考错误原因以及如何修改.
Thought[4]": 函数名错误，应该是intercity_transport_select。重新调用正确的函数。

Repetitive Output Loops

Thought[12]: 我们可以选择黄鹤楼和东湖磨山景区作为主要景点。接下来我们需要查找这两个景点的详细信息。
Action[12]: attractions_select(city="武汉", key="name", func=lambda x: "黄鹤楼" in x or "东湖磨山景区" in x)
…
Thought[13]: 我们可以选择黄鹤楼和东湖磨山景区作为主要景点。接下来我们需要查找这两个景点的详细信息。
Action[13]: attractions_select(city="武汉", key="name", func=lambda x: "黄鹤楼" in x or "东湖磨山景区" in x)
…
Thought[14]: 我们可以选择黄鹤楼和东湖磨山景区作为主要景点。接下来我们需要查找这两个景点的详细信息。
Action[14]: attractions_select(city="武汉", key="name", func=lambda x: "黄鹤楼" in x or "东湖磨山景区" in x)
…
Thought[15]: 我们可以选择黄鹤楼和东湖磨山景区作为主要景点。接下来我们需要查找这两个景点的详细信息。
Action[15]: attractions_select(city="武汉", key="name", func=lambda x: "黄鹤楼" in x or "东湖磨山景区" in x)
…

Reasoning-Action Inconsistency

Thought[2]: "根据我们的经验判断，高铁可能比较快且舒适，可以查看一下高铁的航班。
Action[2]": "intercity_transport_select(start_city="深圳", end_city="南京", intercity_type="airplane")
Observation[2]: ['FlightID' 'From' 'To' 'BeginTime' 'EndTime' 'Duration' 'Cost']\n['FL240' '深圳宝安国际机场' '南京禄口国际机场'…

Thought[14]: 预算超支，我们需要重新规划。我们可以考虑减少一些开销，比如选择更便宜的餐馆或者减少一些景点的游览时间。
Action[14]: planner(query="当前位置广州。我和我朋友想去上海玩一天，预算3100人民币，请给我一个旅行规划。")
Observation[14]: Task finished. The answer is: {"people_number": 2, "start_city": "广州", "target_city": "上海", "itinerary": [{"day": 1, "activities": [{"start_time": "08:00", "end_time": "14:48", "start": "广州南站", "end": "上海虹桥站", "cost": 787.73, "FlightID": "G818", "type": "train", "tickets": 2, "transports": []}, {"position": "外滩", "type": "attraction", "transports": [{"start": "上海虹桥站", "end": "虹桥火车站-地铁站", "mode": "walk" …

Critical Information Missing

Action[15]: planner(query="当前位置广州。我一个人想去武汉玩一天，预算3000人民币，请给我一个旅行规划。")
Observation[15]: Task finished. The answer is: {"people_number": 1, "start_city": "广州", "target_city": "武汉", "itinerary": [{"day": 1, "activities … {"start_time": "18:00", "end_time": "22:39", "start": "武汉站", "end": "广州南站", "cost": 543.6, "TrainID": "G877", "type": "train", "tickets": 1, "transports": []}]}]}"

Action[23]: planner(query="当前位置深圳。我一个人想去苏州玩一天，预算1600人民币，请给我一个旅行规划。")"
Observation[23]: Task finished. The answer is: {"people_number": 1, "start_city": "深圳", "target_city": "苏州", "itinerary": [{"day": 1, "activities": [{"type": "train", "start_time": "06:33", "end_time": "15:46", "cost": 759.35, "tickets": 1, "start": "深圳站", "end": "苏州站", "TrainID": "G2787", "transports": []}, …., "start": "虎丘山风景名胜区", "end": "苏州站", "transports": []}, {"type": "train", "start_time": "22:33", "end_time": "07:00", "cost": 759.35, "tickets": 1, "start": "苏州站", "end": "深圳站", "TrainID": "G2788", "transports": []}]}]}"

Figure 12: Fail case studies of React-one-shot DeepSeek Method.

tination cities of the trip, the number of travel days and constraint information.

3. A detailed example with the query and travel planning response.

Fig. 13 and Fig. 14 respectively show the questionnaire and its translated version.

H.2 Recruitment And Payment

We recruited a total of 250 student volunteers to provide authentic Chinese travel requirements. The participants included 121 undergraduate students, 86 master's students, and 43 doctoral students. The task of understanding the query background and providing travel requirements was estimated to take 1-2 minutes per participant. Given the simplicity of the task and the fact that it did not require extensive professional background or expertise, we compensated each participant with 1 yuan. This compensation was deemed adequate considering the nature of the task and the time required to complete it. The payment was determined based on the estimated time and the straightforward nature of the natural language requirements, ensuring a fair and reasonable reward for the participants.

H.3 Data Consent

When collecting the data, we clearly informed the participants about the usage of the data and the potential irreversible risks of it becoming part of a

	LLMs	DR	EPR		LPR		C-LPR	FPR
			Micro	Macro	Micro	Macro		
Overall Dataset (# 604)								
NeSy Planning		**68.2**	**67.6**	53.6	**68.9**	**54.3**	53.9	44.5
		65.3	67.2	**60.4**	63.8	50.8	**58.2**	**49.0**
		60.0	58.1	31.6	56.1	39.9	26.6	22.6
NeSy Planning* (Oracle Translation)		**71.0**	**69.3**	**59.9**	**69.4**	**62.9**	**58.3**	**59.1**
		60.7	59.7	53.4	60.0	53.3	51.0	50.8
		58.2	57.6	46.5	57.0	49.0	43.2	44.7

Table 13: Experimental results of different LLMs and planning strategies on the overall dataset. LLMs: : DeepSeek-V2.5, : GPT-4o-2024-08-06, :Qwen2.5-7B.

public dataset. We did not track the ID information of the questionnaire respondents. Additionally, we reminded participants not to include any sensitive personal information in the questionnaire responses. During the data cleaning process, we directly removed queries containing offensive content and filtered out sensitive identity information.

H.4 Characteristics Of Annotators

Our data collection process solely involved travel requirements and did not include any protected information, such as sexual orientation or political views as defined under the General Data Protection Regulation (GDPR). All data were collected from native Chinese speakers to ensure that the travel requirements fully align with the context and nuances of the Chinese language. This approach was taken to accurately capture the needs and preferences of the target population, which is primarily composed of Chinese-speaking individuals. The annotators were recruited from a diverse range of academic backgrounds, including undergraduate, master's, and doctoral students, to provide a broad and representative set of travel requirements.

H.5 DSL Annotation for Human Data

The annotation process for the human data involved four stages to ensure the accuracy and validity of the Domain-Specific Language (DSL) annotations: (1) Initial DSL Version Generation: GPT-4o was utilized to provide the initial version of the DSL annotations for the human data. This step aimed to leverage the language model's capabilities to generate a baseline for further refinement. (2) Data Annotation Team Revision: A team of five data annotators was responsible for reviewing and revising the DSL annotations for a total of 250 samples. The team members divided the workload and made necessary corrections to the DSL annotations to ensure their accuracy and relevance to the travel requirements. (3) Primary Developer Verification and Correction: Three of the main developers of the benchmark conducted a thorough review of all the DSL annotations. They verified the correctness of the annotations and made revisions as needed. This stage also involved the exclusion of any invalid queries that could not be verified within the sandbox environment. (4) Final Verification by Primary Developers: The same three main developers performed a final check on all the DSL annotations. This step ensured that the annotations were accurate, consistent, and met the required standards for the benchmark.

Throughout the annotation process, the focus was on ensuring that the DSL annotations accurately captured the travel requirements and were valid within the context of the ChinaTravel benchmark's sandbox environment. The annotation process for human data required a deep understanding of the ChinaTravel DSL and involved joint debugging and verification with the sandbox information. This significantly limited the size of the annotation team, as only a limited number of annotators had the necessary expertise and familiarity with both the DSL and the sandbox environment. Additionally, the process was time-consuming and required meticulous attention to detail, further constraining the rate at which the human dataset could grow. Despite these challenges, the rigorous annotation process ensured the quality and reliability of the human data, which is crucial for the evaluation and development of language agents in real-world travel planning.

开放旅行规划问题搜集

本问卷旨在构建一个开放环境下的旅行规划数据集，以便于相关研究的开展。由于填写的问题将作为公开数据集的一部分，存在无法撤销的风险；请勿在填写内容中包含任何敏感的个人信息，感谢大家的参与！

1. 出发城市：_____（从北京、南京、上海、杭州、深圳、武汉、广州、成都、重庆、苏州中选择）
2. 目标旅游城市：_____（从北京、南京、上海、杭州、深圳、武汉、广州、成都、重庆、苏州中选择）
3. 旅行人数：_____（1-5）
4. 旅行天数：_____（1-5）

您作为用户可以向智能代理发起查询请求。查询内容可以包括对景点、餐饮、住宿、跨城交通(如火车、飞机)以及城内交通(如地铁、步行、出租车)的具体要求。同时，您也可以提供个人偏好。请确保查询中包含以下三个信息:目标城市、人数和天数，并确保这些信息相互匹配。智能代理将根据您的请求提供一个旅行规划结果，包括这几天的交通安排、住宿地点、推荐的景点及餐饮建议。

用户问题的例子:
当前位置苏州。我一个人想去南京玩 2 天，预算 3000 人民币，往返都坐高铁，请给我一个旅行规划。
智能代理回复的例子:
起点:苏州
目的地:南京
交通:苏州北站 -> 南京南站
列车:G4，07:24->08:15
费用:122.9 元
车票:1 张
游览:玄武湖景区
交通:地铁(南京南站 ->南京林业大学・新庄)，步行 3 分钟 +地铁 23 分钟+步行 8 分钟
费用:4 元
游览时间:08:50->10:00
门票:0 元
……
午餐:南京金鹰国际酒店・满园春中餐厅
费用:188 元
时间:12:10 ->13:10
住宿:桔子水晶南京玄武湖酒店
房型:大床房，1 间
费用:370 元
返回:南京南站 > 苏州站
列车:G7220，20:09->21:23
费用:122.9 元
车票:1 张

我们将用户问题分为不同难度级别进行分类，以下是每个级别的描述
低级:涉及一般性问题，不包含个性化需求。
中级:包含一定程度的个性化需求，通常涉及到食宿交通等方面。
高级:涉及更复杂、更具体的需求，如时间要求、特定地点或活动的安排等。

以下是不同难度级别下的用户问题示例:
低级:我想知道去上海玩 2 天的行程规划，从杭州出发。
中级:我想独自一人前往南京穷游，计划在那里待 3 天左右。我对历史文化很感兴趣，希望能深度游览一些古迹。
高级:我们三人后天需要前往北京玩 2 天。第二天晚上十点前需要从北京站返回。想在第一天去故宫，第二天去天坛，请给一个旅行规划

5. 请给出用户问题：____

Figure 13: Questionnaire

Open Travel Planning Data Collection Questionnaire

This questionnaire aims to construct a dataset for travel planning in an open environment to facilitate relevant research. Since the responses will be part of a public dataset and cannot be revoked, please do not include any sensitive personal information in your responses. Thank you for your participation!

1. Departure City:_____ (Choose from Beijing, Nanjing, Shanghai, Hangzhou, Shenzhen, Wuhan, Guangzhou, Chengdu, Chongqing, Suzhou)
2. Destination City: _____ (Choose from Beijing, Nanjing, Shanghai, Hangzhou, Shenzhen, Wuhan, Guangzhou, Chengdu, Chongqing, Suzhou)
3. Number of Travelers:_____ (1-5)
4. Number of Travel Days:_____ (1-5)

As a user, you can submit queries to the intelligent agent. Your query may include specific requirements for attractions, dining, accommodation, intercity transportation (e.g., train, plane), and intra-city transportation (e.g., subway, walking, taxi). You may also provide personal preferences. Please ensure that your query includes the following three pieces of information: the destination city, the number of travelers, and the number of travel days, and make sure they are consistent. The intelligent agent will generate a travel plan based on your request, covering transportation arrangements, accommodation, recommended attractions, and dining suggestions.

Example User Query:
"My current location is Suzhou. I want to travel alone to Nanjing for 2 days with a budget of 3,000 RMB, taking the high-speed train for both departure and return. Please provide a travel plan."

Example Response from the Intelligent Agent:

Departure: Suzhou
Destination: Nanjing
Transportation: Suzhou North Station → Nanjing South Station
Train: G4, 07:24 → 08:15
Cost: 122.9 RMB
Tickets: 1
Attraction: Xuanwu Lake Scenic Area
Transportation: Subway (Nanjing South Station → Nanjing Forestry University·Xinzhuang)
Route: Walk 3 minutes → Subway 23 minutes → Walk 8 minutes
Cost: 4 RMB
Visit Time: 08:50 → 10:00
Admission: 0 RMB
…
Lunch: Nanjing Jinling Hotel · Man Yuan Chun Chinese Restaurant
Cost: 188 RMB
Time: 12:10 → 13:10
Accommodation: Crystal Orange Hotel Nanjing Xuanwu Lake
Room Type: Queen Room, 1 room
Cost: 370 RMB
Return: Nanjing South Station → Suzhou Station
Train: G7220, 20:09 → 21:23
Cost: 122.9 RMB
Tickets: 1
Classification of User Queries by Difficulty Level
We categorize user queries into different difficulty levels as follows:

Easy Level: General inquiries without personalized requirements.
Medium Level: Includes some degree of personalization, usually involving food, lodging, or transportation.
Hard Level: Involves more complex and specific needs, such as time constraints, particular locations, or planned activities.
Examples of User Queries at Different Difficulty Levels:
Basic Level: "I want to know the itinerary for a 2-day trip to Shanghai from Hangzhou."
Intermediate Level: "I plan to travel alone to Nanjing on a budget and stay for about three days. I'm interested in history and culture and would like to explore historical sites in depth."
Advanced Level: "Three of us need to travel to Beijing the day after tomorrow for a 2-day trip. We need to return from Beijing Railway Station before 10 PM on the second day. We want to visit the Forbidden City on the first day and the Temple of Heaven on the second day. Please provide a travel plan."

5. Please provide a user query:____

Figure 14: The translated version of the questionnaire

ChinaTravel	TravelPlanner
当前位置武汉。我一个人想去苏州玩一天，预算 1400 人民币，请给我一个旅行规划。 Current location: Wuhan. I want to visit Suzhou for a day by myself with a budget of 1,400 RMB. Please provide me with a travel plan.	Please help me plan a trip from St. Petersburg to Rockford spanning 3 days from March 16th to March 18th, 2022. The travel should be planned for a single person with a budget of $1,700.
当前位置南京。我一个人想去重庆玩 3 天，喜欢吃甜食面包啥的，请给我一个旅行规划。 Current location: Nanjing. I want to travel to Chongqing alone for 3 days. I like sweet foods and bread. Please provide me with a travel plan.	Please design a travel plan departing from Las Vegas and heading to Stockton for 3 days, from March 3rd to March 5th, 2022, for one person, with a budget of $1,400.
当前位置重庆。我和朋友两个人想去武汉玩 3 天，想尝试当地菜，请给我们一个旅行规划。 Current location: Chongqing. My friend and I want to visit Wuhan for 3 days and try the local cuisine. Could you please provide us with a travel plan?	Craft a travel plan for me to depart from New Orleans and head to Louisville for 3 days, from March 12th to March 14th, 2022. I will be travelling alone with a budget of $1,900.
当前位置成都。我们三个人想去深圳玩 2 天，想去历史感比较重的景点，请给我们一个旅行规划。 Current location: Chengdu. The three of us want to visit Shenzhen for 2 days and are interested in historical sites. Could you please provide us with a travel itinerary?	Could you aid in curating a 5-day travel plan for one person beginning in Denver and planning to visit 2 cities in Washington from March 23rd to March 27th, 2022? The budget for this trip is now set at $4,200.
当前位置深圳。我和朋友两个人想去上海玩 3 天，想去海洋水族馆，请给我们一个旅行规划。 Current location: Shenzhen. My friend and I want to visit Shanghai for 3 days and we would like to go to the Ocean Aquarium. Could you please provide us with a travel plan?	Could you assist in crafting a travel itinerary for a 5-day, single-person trip departing from Orlando and touring 2 cities in Texas? The travel dates should range from March 10th to March 14th, 2022, and the entire travel budget is $3,100.
当前位置成都。我和朋友两个人想去上海玩 3 天，住一间双床房，期间可能要开会，酒店最好能提供个开会的地方，请给我一个旅行规划。 Current location: Chengdu. My friend and I want to visit Shanghai for 3 days. We need a twin room, and we might need a meeting space during our stay. Please provide me with a travel plan.	Could you help me arrange a 7-day solo travel itinerary from Kona to California with a budget of $5,800, intending to visit 3 distinct cities in California from March 7th to March 13th, 2022?
我目前在南京，计划和两个朋友一起去上海玩两天，选择原舍 • 在水一方度假酒店，请帮我们规划一个旅行方案。 I am currently in Nanjing and plan to travel to Shanghai with two friends for two days. We have chosen the YuanShe · Zai Shui Yi Fang Resort Hotel. Please help us plan a travel itinerary.	Please help me craft a 7-day travel plan. I'm planning on leaving from Punta Gorda and exploring 3 different cities in Wisconsin from March 16th to March 22nd, 2022. The budget for this trip is set at $5,700.
当前位置北京。我和三个朋友计划去成都玩两天，选择火车出行，市内交通方式为地铁。请给我一个旅行规划。 Current location: Beijing. My three friends and I are planning to visit Chengdu for two days. We have chosen to travel by train and use subway for city transportation. Please provide me with a travel itinerary.	Could you help me create a 7-day travel plan starting on March 18th, 2022, and ending on March 24th, 2022? The trip will start in Washington and I would like to visit 3 cities in Minnesota. This trip is for one person with a budget of $7,200.

Figure 15: Examples of easy-level queries from ChinaTravel and TravelPlanner.

ChinaTravel	TravelPlanner
当前位置武汉。我两个人想去苏州玩 2 天，预算 4000 人民币，坐火车去，住一间大床房，想去虎丘山风景名胜区这样的自然风光，请给我一个旅行规划。 Current location: Wuhan. Two of us want to visit Suzhou for 2 days with a budget of 4000 RMB. We plan to take the train and stay in a room with a king-size bed. We would like to visit natural attractions like Tiger Hill Scenic Area. Please provide a travel itinerary.	Could you please arrange a 3-day trip for two, starting in Sacramento and heading to Atlanta, from March 14th to March 16th, 2022. The budget for this trip is $4,700, and we require accommodations where parties are allowed.
当前位置广州。我两个人想去成都玩 3 天，预算 9000 人民币，坐火车往返，住一间大床房，麻烦给我一个旅行规划。 Current location: Guangzhou. Two of us want to visit Chengdu for 3 days with a budget of 9,000 RMB. We plan to travel round-trip by train and stay in a room with a double bed. Could you please provide a travel itinerary for us?	Could you please design a 3-day travel plan for a group of 5, departing from Manchester and heading to Charlotte, from March 29th to March 31st, 2022? Our budget is set at $4,800 and we would prefer to have entire rooms for our accommodations.
当前位置广州。我和我的两个朋友想去深圳玩两天，预算 2100 人民币，住两间双床房，坐地铁游玩，想吃海鲜，想去深圳欢乐谷玩。Current location: Guangzhou. My two friends and I want to go to Shenzhen for two days. Our budget is 2,100 RMB. We plan to stay in two twin-bed rooms, travel around by metro, eat seafood, and visit Shenzhen Happy Valley.	Could you tailor a 5-day travel plan for two people, departing from Knoxville and visiting 2 cities in Florida from March 20 to March 24, 2022? Our budget is set at $3,900. We'd love to explore local Chinese and Mediterranean cuisines during our stay.
当前位置武汉。我两个人想去杭州玩 3 天，预算 7000 人民币，坐飞机往返，住一间大床房，麻烦给我一个旅行规划。 Current location: Wuhan. Two of us want to visit Hangzhou for 3 days with a budget of 7,000 RMB. We plan to travel by plane round-trip and stay in a room with a large bed. Could you please provide a travel plan for us?	Could you help create a 7-day travel plan for a group of 3, departing from Greensboro and touring 3 different cities in Georgia from March 10th to March 16th, 2022? We have a new budget of $4,000 for this trip. We'd also appreciate if our accommodations have smoking areas.
当前位置杭州。我两个人想去苏州玩 2 天，预算 3500 人民币，住一间大床房，想去看看拙政园这样的园林景观，请给我一个旅行规划。 Current location: Hangzhou. Two of us want to visit Suzhou for 2 days with a budget of 3,500 RMB. We would like to stay in a room with a large bed and visit garden attractions like the Humble Administrator's Garden. Please provide a travel plan.	Could you help create a 5-day travel itinerary for a group of 4, starting from New York and visiting 2 cities in Louisiana from March 15th to March 19th, 2022? We have a budget of $12,300. Please note that we require accommodations where smoking is permissible.
当前位置北京。我两个人想去深圳玩 3 天，预算 7000 人民币，住一间大床房，坐飞机去，酒店最好有泳池，想去深圳欢乐谷看一下，请给我一个旅行规划。 Current location: Beijing. Two of us want to visit Shenzhen for 3 days with a budget of 7,000 RMB. We would like to stay in a hotel with a king-size bed and preferably a swimming pool. We plan to fly there and would like to visit Shenzhen Happy Valley. Please provide a travel itinerary.	Can you provide me with a 5-day travel plan for 2 people, starting from Asheville and exploring 2 cities in New York from March 13th to March 17th, 2022? Our budget is set at $4,700 and we would love to try local Mexican and Chinese cuisines during our trip.

Figure 16: Examples of medium-level queries from ChinaTravel and TravelPlanner.

ChinaTravel	TravelPlanner
[当前位置武汉,目标位置南京,旅行人数 2,旅行天数 4] 我和同学 2 人打算去南京玩 4 天，预算 1500 (不包括车票住宿)，只是玩和吃饭，请你帮忙规划。 [Current location: Wuhan, Destination: Nanjing, Number of travelers: 2, Duration of travel: 4 days] My classmate and I are planning to visit Nanjing for 4 days. Our budget is 1500 (excluding transportation and accommodation), just for activities and meals. Please help us plan.	Can you create a 5-day itinerary for a group of 7 people traveling from Richmond to two cities in Florida between March 9th and 13th, 2022? Our budget is $8,500. We require accommodations that allow visitors and should ideally be entire rooms. In regards to dining options, we prefer French, American, Mediterranean, and Italian cuisines.
[当前位置南京,目标位置成都,旅行人数 3,旅行天数 5] 我们一家三口想去成都旅游一周，主要想逛一些适合带小朋友的景点，预算 8000 元，然后品尝一些当地的美食。 [Current location: Nanjing, Destination: Chengdu, Number of travelers: 3, Travel days: 5] Our family of three wants to travel to Chengdu for a week. We mainly want to visit attractions suitable for children, with a budget of 8,000 yuan, and also taste some local delicacies.	Could you help design a travel plan for two people leaving from Houston to Pensacola for 3 days, from March 6th to March 8th, 2022? Our budget is set at $1,400 for this trip and we require our accommodations to be visitor-friendly. We would like to have options to dine at Indian, American, Chinese, and Italian restaurants. We also prefer not to self-drive during the trip.
[当前位置广州,目标位置深圳,旅行人数 3,旅行天数 2] 我们一行三人要从广州去到深圳玩两天，想去繁华的街区逛逛，尽可能减少麻烦的交通，总消费尽可能少。 [Current location: Guangzhou, Destination: Shenzhen, Number of travelers: 3, Number of travel days: 2] Our group of three plans to travel from Guangzhou to Shenzhen for two days. We want to explore bustling neighborhoods, minimize inconvenient transportation, and keep the total expenses as low as possible.	Could you help create a 3-day travel plan for two people? We're traveling from West Palm Beach to White Plains, visiting only one city from March 5th to March 7th, 2022. We have a budget of $2,600. For our accommodations, we'd like rooms that are not shared. We are not planning on self-driving and will be reliant on public transportation. Cuisines we are interested in trying include Mexican, Chinese, Mediterranean, and American.
[当前位置苏州,目标位置杭州,旅行人数 4,旅行天数 2] 我想 4 个人去杭州 2 天进行历史文化遗址的考察顺带玩一下。 [Current location: Suzhou, Destination: Hangzhou, Number of travelers: 4, Duration of travel: 2 days] I would like 4 people to go to Hangzhou for 2 days to explore historical and cultural sites and have some fun along the way.	Could you generate a 3-day travel plan for a group of 3 people, departing from Bangor and visiting Washington from March 21st to March 23rd, 2022? Our budget is set at $3,100. We require accommodations that are pet-friendly and we would prefer to have entire rooms to ourselves. We do not plan on self-driving for this trip
[当前位置上海,目标位置北京,旅行人数 1,旅行天数 3] 我要从上海出发，到北京玩三天，希望看一些名胜古迹，吃一些当地特色，预算充分。 [Current location: Shanghai, Destination: Beijing, Number of travelers: 1, Number of travel days: 3] I want to depart from Shanghai and spend three days in Beijing. I hope to see some famous landmarks and try some local specialties, with a sufficient budget.	Could you help with creating a 5-day travel plan for 2 people, originating from Evansville and covering 2 cities in Texas from March 17th to March 21st, 2022? Our preferred accommodations are private rooms, and they must permit children under 10 since we will have them with us. Transportation should not involve any flights. The budget for this trip is set at $2,800.
[当前位置北京,目标位置上海,旅行人数 2,旅行天数 3] 我和朋友计划用三天的时间从北京到上海玩，计划坐飞机来回，偏红色旅游线路。 [Current location: Beijing, Destination: Shanghai, Number of travelers: 2, Number of travel days: 3] My friend and I are planning to spend three days traveling from Beijing to Shanghai. We plan to fly round trip and prefer a red-themed travel route.	Can you assist in creating a travel itinerary for a group of 4, starting in Seattle and visiting 3 unique cities across Texas? This trip will span over 7 days from March 10th through March 16th, 2022. We have a budget of $11,000. Regarding our accommodations, we would like to rent entire rooms, and it's important that our lodgings allow parties. As for transportation, we do not plan to drive ourselves around.

Figure 17: Examples of human/hard level queries from ChinaTravel and TravelPlanner.

Prompts for POI recommendation

```
NEXT_POI_TYPE_INSTRUCTION = """
   You are a travel planning assistant.
   The user's requirements are: {}.
   Current travel plans are: {}.
   Today is {}, current time is {}, current location is {}, and
      POI_type_list is {}.
   Select the next POI type based on the user's needs and the
      current itinerary.
   Please answer in the following format.
   Thought: [Your reason]
   Type: [type in POI_type_list]
    """
```

Figure 18: Prompts for next-POI-type recommendation

Prompts for restaurants recommendation

```
RESTAURANT_RANKING_INSTRUCTION = """
    You are a travel planning assistant.
    The user's requirements are: {user_requirements}.
    The restaurant info is:
    {restaurant_info}
    The past cost for intercity transportation and hotel
       accommodations is: {past_cost}.

    Your task is to select and rank restaurants based on the
       user's needs and the provided restaurant information.
       Consider the following factors:
    1. Restaurant name
    2. Cuisine type
    3. Price range
    4. Recommended food

    Additionally, keep in mind that the user's budget is
       allocated across multiple expenses, including intercity
       transportation and hotel accommodations. Ensure that the
       restaurant recommendations fit within the remaining
       budget constraints after accounting for the past cost.
    Note that the price range provided for each restaurant is
       the average cost per person per meal, the remaining
       budget must cover the cost of three meals per day for {
       days} days.

    For each day, recommend at least 6 restaurants, combining
       restaurants for all days together.

    Your response should follow this format:

    Thought: [Your reasoning for ranking the restaurants]
    RestaurantNameList: [List of restaurant names ranked by
       preference, formatted as a Python list]
    """
```

Figure 19: Prompts for restaurant recommendation

Prompts for attractions recommendation

```
ATTRACTION_RANKING_INSTRUCTION = """
    You are a travel planning assistant.
    The user's requirements are: {user_requirements}.
    The attraction info is:
    {attraction_info}
    The past cost for intercity transportation and hotel
       accommodations is: {past_cost}.

    Your task is to select and rank attractions based on the
       user's needs and the provided attraction information.
       Consider the following factors:
    1. Attraction name
    2. Attraction type
    3. Location
    4. Recommended duration

    Additionally, keep in mind that the user's budget is
       allocated across multiple expenses, including intercity
       transportation and hotel accommodations. Ensure that the
       attraction recommendations fit within the remaining
       budget constraints after accounting for the past cost.

    For each day, recommend at least 8 attractions, combining
       attractions for all days together. To ensure a
       comprehensive list, consider a larger pool of candidates
       and prioritize diversity in attraction type and location.

    Your response should follow this format:

    Thought: [Your reasoning for ranking the attractions]
    AttractionNameList: [List of attraction names ranked by
       preference, formatted as a Python list]

    Example:
    Thought: Based on the user's preference for historical sites
        and natural attractions, the attractions are ranked as
       follows:
    AttractionNameList: ["Attraction1", "Attraction2", ...]
    """
```

Figure 20: Prompts for attraction recommendation