PFDial: A Structured Dialogue Instruction Fine-tuning Method Based on UML Flowcharts

Ming Zhang[1*], **Yuhui Wang**[1*], **Yujiong Shen**[1*], **Tingyi Yang**[1], **Changhao Jiang**[1], **Yilong Wu**[1], **Shihan Dou**[1], **Qinhao Chen**[5], **Zhiheng Xi**[1], **Zhihao Zhang**[1], **Yi Dong**[1], **Zhen Wang**[2], **Zhihui Fei**[2], **Mingyang Wan**[2], **Tao Liang**[2], **Guojun Ma**[2†], **Qi Zhang**[1,4†], **Tao Gui**[3,4], **Xuanjing Huang**[1,3,4]

[1] College of Computer Science and Artificial Intelligence, Fudan University
[2] Douyin Co., Ltd.
[3] Institute of Modern Languages and Linguistics, Fudan University
[4] Institute of Trustworthy Embodied Artificial Intelligence, Fudan University
[5] Graduate School of Arts and Sciences, Columbia University

mingzhang23@m.fudan.edu.cn
maguojun@bytedance.com
qz@fudan.edu.cn

Abstract

Process-driven dialogue systems, which operate under strict predefined process constraints, are essential in customer service and equipment maintenance scenarios. Although Large Language Models (LLMs) have shown remarkable progress in dialogue and reasoning, they still struggle to solve these strictly constrained dialogue tasks. To address this challenge, we construct **P**rocess **F**low **Dial**ogue (**PFDial**) dataset, which contains 12,705 high-quality Chinese dialogue instructions derived from 440 flowcharts containing 5,055 process nodes. Based on PlantUML specification, each UML flowchart is converted into atomic dialogue units i.e., structured five-tuples. Experimental results demonstrate that a 7B model trained with merely 800 samples, and a 0.5B model trained on total data both can surpass 90% accuracy. Additionally, the 8B model can surpass GPT-4o up to 43.88% with an average of 11.00%. We further evaluate models' performance on challenging backward transitions in process flows and conduct an in-depth analysis of various dataset formats to reveal their impact on model performance in handling decision and sequential branches. The data is released in https://github.com/KongLongGeFDU/PFDial.

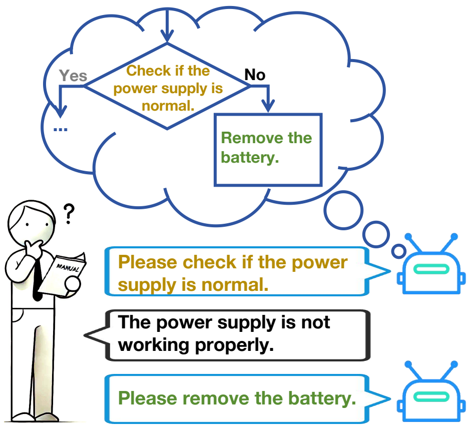

Check if the power supply is normal.
Yes
No
...
Remove the battery.
?
MANUAL
Please check if the power supply is normal.
The power supply is not working properly.
Please remove the battery.

Figure 1: In this scenario, the system interacts with the user based on a flowchart that checks whether the power supply is functioning properly. If the power supply is faulty, the system guides the user to remove the battery. This interaction follows the decision-making process outlined in the flowchart.

1 Introduction

Process-driven dialogue systems (Yi et al., 2024), as a special type of task-oriented dialogue systems, play a crucial role in various real-world applications, particularly in scenarios such as customer service, equipment maintenance, and medical consultation, where strict adherence to predefined process constraints is essential. In these contexts, dialogue systems must navigate through complex decision trees while maintaining precise control over the conversation flow, ensuring both compliance with established procedures and effective user interaction. These process flows typically contain two types of branch: sequential branches that follow a linear progression and decision branches that require conditional routing based on user input. Figure 1 illustrates an example scenario of process-driven dialogue systems.

The emergence of Large Language Models (LLMs) has brought unprecedented capabilities in natural language understanding and generation, demonstrating remarkable performance in both dialogue and reasoning tasks. These advances suggest

* Equal Contribution.
† Corresponding Author.

potential solutions for process-driven dialogue systems (Yi et al., 2024; Zhang et al., 2023; Wu et al., 2020). However, our empirical evaluation reveals that even state-of-the-art (SOTA) LLMs such as GPT-4o (OpenAI, 2023) struggle to consistently maintain process constraints while engaging in dialogue. Specifically, these models often deviate from predefined process constraints, make incorrect state transitions, or fail to properly handle complex decision branches, highlighting the need for more specialized solutions.

To address this challenge, we construct **P**rocess **F**low **Dial**ogue (PFDial) dataset, which contains 12,705 high-quality dialogue instructions derived from 440 flowcharts containing 5,055 process nodes. Based on PlantUML specification, each UML flowchart is converted into atomic dialogue units, forming structured five-tuples (flowchart description, current state, user input, next state, robot output). This structured representation enables models to learn precise state transitions while maintaining natural dialogue capabilities. Through supervised fine-tuning (SFT) on PFDial, models can acquire strong controlled reasoning capabilities for process flows effectively following the prescribed state transitions and decision logic.

We conducted comprehensive experiments to address four key questions:

(1) How does our approach perform compared to SOTA LLMs? Our main experimental results demonstrate that models with varying parameter sizes can achieve excellent results after SFT on total data of PFDial. For instance, even a 0.5B model can achieve accuracy of 98.99% and 92.79% on in-domain and out-of-domain tests, respectively. More impressively, an 8B model achieves 97.02% accuracy on out-of-domain tests, surpassing GPT-4o by 11.00%, with over 43.88% improvement in decision branches.

(2) How does model performance scale with training data size? Our data scaling experimental results show that a 7B model can surpass 90% accuracy with only 800 training samples. As the training dataset increases, the overall performance of the model continues to improve. This underscores PFDial's exceptional effectiveness in enhancing the model's capability for controlled reasoning tasks with minimal data.

(3) How effective is our approach in handling backward transitions? We evaluated the model's performance on more challenging backward transitions in decision branches using our constructed dataset, PFDial-H. This specialized benchmark focuses on cases where the next state returns to a previous point in the process flow. These transitions are particularly challenging due to their relative scarcity and the complex reasoning they require. Results on PFDial-H further validate our approach's superiority in challenging controlled reasoning tasks.

(4) How do different state representation formats affect model performance? Through systematic analysis of three different dataset formats, we provide interpretable insights into how different formats impact model performance on decision and sequential branches, offering valuable guidance for future research.

Overall, our contributions can be summarized as follows:

- We have developed the Process Flow Dialogue (PFDial) dataset, which is derived from 440 flowcharts encompassing 5,055 process nodes. This dataset contains 12,705 high-quality dialogue instructions, serving as a valuable resource for training process-driven dialogue systems.

- The comprehensive experiments demonstrate that the PFDial dataset is highly effective. Even models with a smaller number of parameters (e.g., 0.5B, 1B, 1.5B) or those trained on relatively limited data (800 training samples) can achieve high accuracy. An 8B model achieves 97.02% accuracy on out-of-domain tests, surpassing GPT-4o by 11.00%, with over 43.88% improvement in decision branches.

- We demonstrate our model's superior performance on complex backward transitions in decision branches using the PFDial-H benchmark, highlighting its capability in handling rare and intricate reasoning tasks. Additionally, we provide insights into the impact of different dataset formats on model performance, offering guidance for future research.

2 PFDial

In this section, we introduce PFDial, a Chinese dataset specifically designed for process-driven dialogue systems.

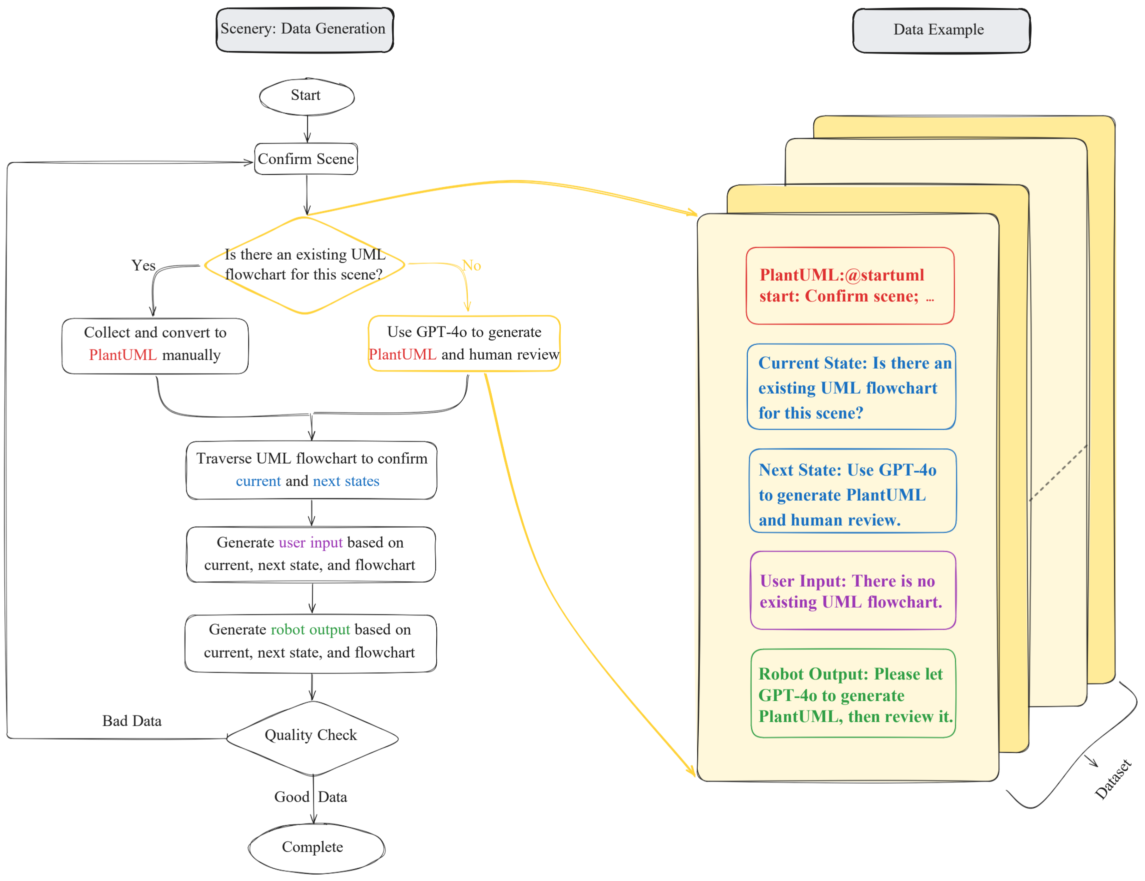

Scenery: Data Generation
Start
Confirm Scene
Is there an existing UML flowchart for this scene?
Yes
No
Collect and convert to PlantUML manually
Use GPT-4o to generate PlantUML and human review
Traverse UML flowchart to confirm current and next states
Generate user input based on current, next state, and flowchart
Generate robot output based on current, next state, and flowchart
Bad Data
Quality Check
Good Data
Complete
Data Example
PlantUML:@startuml start: Confirm scene; ...
Current State: Is there an existing UML flowchart for this scene?
Next State: Use GPT-4o to generate PlantUML and human review.
User Input: There is no existing UML flowchart.
Robot Output: Please let GPT-4o to generate PlantUML, then review it.
Dataset

Figure 2: The left side illustrates the data construction process. The right side shows an example of the five-tuple dataset generated based on the leftside flowchart.

Statistics	**Train**	**ID Test**	**OOD Test**
Flowcharts	440	80	80
State Nodes	5055	902	1262
Sequential Samples	9029	1628	2265
Decision Samples	3676	645	698
Dialogue Samples	12705	2273	2963
Avg. Length	277.16	270.57	326.10

Table 1: Statistics of the PFDial Dataset

2.1 Dataset Overview

PFDial The construction of PFDial follows a systematic pipeline, including flow chart collection, textual representation conversion, state transition Information extraction, prompt generation, and data validation. The dataset combines real-world scenarios and synthetic data, achieving broad coverage of practical applications while maintaining high quality and diversity. Table 1 presents detailed statistics of the PFDial dataset.

PFDial-H Considering the prevalence and importance of backward transitions in practical applications, we specifically constructed a supplementary dataset that incorporates backward transition mechanisms called PFDial-Hard (PFDial-H). By strategically adding backward transition nodes to existing flowcharts using GPT-4o, we implemented backward transition functionality for cases where conditions are not met. This improvement was applied to both the out-of-domain test and training flowcharts, generating new training samples through the same prompt augmentation process. Table 5 in Appendix B.2 presents detailed statistics of the PFDial-H dataset.

2.2 Dataset Construction Process

Flow Chart Collection Through extensive research, we identified and categorized 90 specific business scenarios, details of which can be found in the Appendix 18. Additionally, we designed a carefully constructed template dataset. After identifying the business scenarios, we collected the flowcharts manually or automatically, depending on whether pre-existing flowcharts were available. This process, combining automation with human review, significantly improved the efficiency and accuracy of UML flowchart generation.

Textual Representation Conversion To efficiently convert flowcharts into machine-readable formats, we conducted a comparative study of several text-based representation schemes, including PlantUML[1], chart-mage[2], nomnoml[3], and Mermaid[4]. After comprehensive evaluation, we selected PlantUML as the standard format. This decision was made for several reasons: First, PlantUML employs a code-like structured representation utilizing syntax features such as indentation, branching, and loops, making flowchart descriptions both intuitive to read and convenient for program processing. Second, a preliminary experiment using GPT-4o demonstrated that the PlantUML format exhibited superior accuracy compared to other approaches. Details can be find in Table 4 in Appendix B.1. During data processing, we represented all flowcharts in PlantUML format and generated standardized state nodes and their transition relationships through parsing, providing a unified representation for subsequent five-tuple dataset generation.

State Transition Information Extraction Based on the standardized PlantUML representation, we developed a specialized algorithm to extract complete state transition information, which is shown in Algorithm 1 in Appendix A.1. During this process, we got all existing paths and identified 5,055 distinct state nodes from the training set. Each state transition pair (current state -> next state) strictly corresponds to a specific path in the flowchart, ensuring the accuracy and consistency of the state information extraction.

Prompt Generation To create quality training samples, we used the GPT-4o model for bidirectional prompt augmentation for each state transition. This involved generating user input and robot output based on the current and next states, along with the flowchart. Detailed prompts can be found in Appendix D.

Data Validation To ensure the reliability of the dataset, we implemented a rigorous multi-level validation process: first, ensuring that all state nodes strictly correspond to the original flowcharts; second, validating the syntax correctness of the PlantUML; and finally, checking the logical consistency between user inputs and state transitions. Any data that does not meet these criteria will be regenerated.

This dataset construction methodology not only ensures data quality and diversity, but also provides a solid foundation for subsequent model training. Through systematic construction processes and strict quality control, the PFDial dataset effectively balances authenticity, standardization, and scalability requirements. Figure 2 illustrates the data construction process flowchart and an example of the five-tuple dataset generated based on the data construction flowchart.

3 Experiments

In this chapter, we present a series of comprehensive experiments to address the four key questions mentioned in Section 1. We conducted the main experiment, data scaling experiment, backward transition studies, and format ablation studies respectively.

3.1 Experimental Setup

Base Models We evaluate our method using two series of base models with varying parameter sizes. The first series includes Qwen2.5 models (Yang et al., 2024) ranging from 0.5B to 7B parameters (Qwen2.5-0.5B, Qwen2.5-1.5B, Qwen2.5-3B, and Qwen2.5-7B). The second series consists of Llama3 models (Dubey et al., 2024) spanning from 1B to 8B parameters (Llama3.2-1B, Llama3.2-3B, and Llama3.1-8B). This diverse selection of models enables us to comprehensively analyze the impact of model scale on performance.

Baselines For comparison, we select a comprehensive set of both open-source and proprietary state-of-the-art (SOTA) LLMs that have demonstrated strong performance across various NLP tasks. These include proprietary models like GPT-4o (OpenAI, 2023), GPT-3.5-turbo (Ouyang et al., 2022a), Gemini-2.0-flash-exp (Anil et al., 2023), and Claude-3.5-sonnet [5], as well as open-source models such as DeepSeek-v3 (DeepSeek-AI et al., 2024), Llama3.1-8b-instruct (Dubey et al., 2024), and Qwen2.5-7b-instruct (Yang et al., 2024). These models represent the current frontier of language model capabilities and serve as strong baselines for evaluating our approach.

[1]https://plantuml.com/
[2]https://chartmage.com/intro.html
[3]https://www.nomnoml.com/
[4]https://mermaid.js.org/

[5]https://www.anthropic.com/news/claude-3-family

Model	ID-test			OOD-test		
	Acc	Decision Acc	Sequential Acc	Acc	Decision Acc	Sequential Acc
Baselines						
LLaMA-3.1-8B-Instruct	–	–	–	13.20	2.16	15.00
Claude-3.5-Sonnet	–	–	–	62.74	22.06	69.40
Qwen2.5-7B-Instruct	–	–	–	65.87	37.88	71.34
Gemini-2.0-Flash-Exp	–	–	–	75.17	47.48	79.66
DeepSeek-v3	–	–	–	79.02	47.72	84.11
GPT-3.5-Turbo	–	–	–	79.76	39.57	86.29
GPT-4o	–	–	–	86.29	51.80	91.90
FineTuned on PFDial						
LLaMA-3.2-1B	98.90	**98.59**	98.96	93.57	87.05	94.62
LLaMA-3.2-3B	98.77	98.02	98.91	95.81	91.37	96.53
LLaMA-3.1-8B	**99.03**	98.31	**99.17**	**97.29**	**96.88**	97.35
Qwen2.5-0.5B	98.99	98.02	**99.17**	91.35	89.45	91.66
Qwen2.5-1.5B	98.90	97.46	**99.17**	94.00	88.97	94.82
Qwen2.5-3B	98.77	98.31	98.85	94.97	89.69	95.84
Qwen2.5-7B	98.94	98.02	99.11	96.51	90.65	**97.47**

Table 2: Results on PFDial: Decision Acc represents the accuracy of the decision branch, and Sequential Acc reflects the accuracy of the sequential branch. Acc is The overall accuracy.

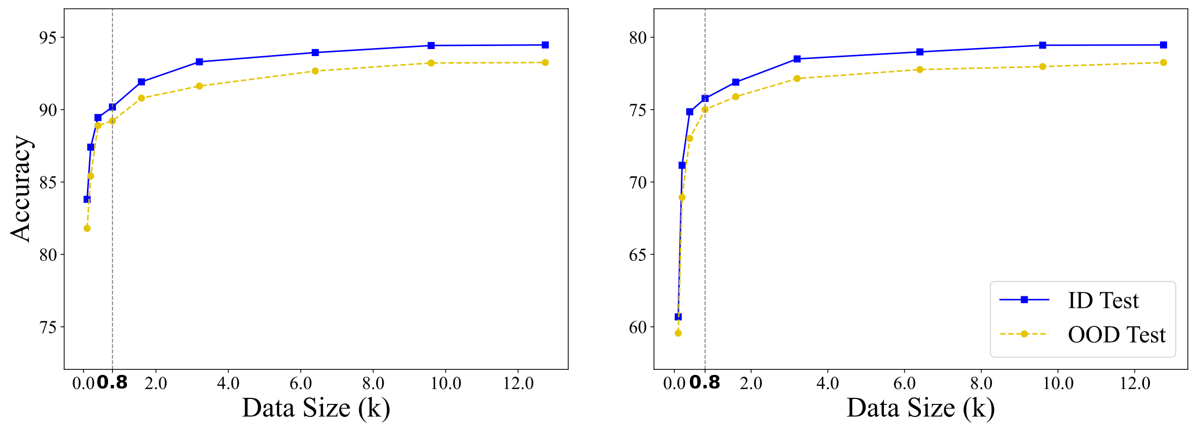


Figure 3: Results on Data Scaling: The left plot shows the accuracy of ID and OOD tests using data scaling strategy **(a)**, while the right plot shows the accuracy using data scaling strategy **(b)**.

Datasets Our main experiments utilize the PFDial dataset containing 12,705 training samples. For evaluation, we maintain two test sets: our out-of-domain test set comprising 698 decision branches and 2,963 sequential branches, and an in-domain test set containing 645 decision branches and 1628 sequential branches. For backward transition experiments, we use the specially constructed PFDial-H dataset. For format ablation studies, we construct corresponding training and test sets in three different state representation formats to systematically evaluate their impact on model performance.

Implementation Details We combine the PFDial dataset with the general dialogue dataset BELLE (BELLEGroup, 2023; Wen et al., 2023) in a 1:1 ratio for SFT. For the training process, we utilize the OpenRLHF(Hu et al., 2024) framework. For instance, the training of Qwen2.5-7B model is conducted on 8 H20 GPUs, with a total training time of approximately 1 hour. For detailed hyperparameter configurations, please refer to Table 6 in Appendix C.1.

Evaluation Metrics Model performance is evaluated by exact match accuracy between predicted and ground truth next states. For a prediction to be considered correct, it must exactly match the ground truth state transition. Specifically, our dataset contains two types of samples: sequential samples and decision samples. Sequential samples refer to cases where there is only one unique next state given the current state. Decision samples

Model	PFDial-H			OOD-test of PFDial		
	Acc	Backward Acc(Dist <5)	Backward Acc(Dist ≥5)	Acc	Decision Acc	Sequential Acc
Baselines						
LLaMA-3.1-8B-Instruct	15.00	13.64	15.52	13.20	2.16	15.00
Claude-3.5-Sonnet	57.50	59.09	56.90	62.74	22.06	69.40
Qwen2.5-7B-Instruct	31.25	31.82	31.03	65.87	37.88	71.34
Gemini-2.0-Flash-Exp	26.25	22.73	27.59	75.17	47.48	79.66
DeepSeek-v3	40.00	31.82	43.10	79.02	47.72	84.11
GPT-3.5-Turbo	51.25	54.55	50.00	79.76	39.57	86.29
GPT-4o	58.75	68.18	55.17	86.29	51.80	91.90
Secondary Fine-tuning						
Qwen2.5-0.5B	58.75	77.27	51.72	50.77	52.67	39.09
Qwen2.5-1.5B	47.50	45.45	48.28	87.90	88.82	82.25
Qwen2.5-3B	57.50	59.09	56.90	79.73	80.91	72.42
Qwen2.5-7B	66.25	90.91	56.90	76.04	76.82	71.22
Integrated Training						
Qwen2.5-0.5B	65.00	72.73	62.07	92.76	93.22	89.93
Qwen2.5-1.5B	70.00	72.73	68.97	93.87	94.82	88.01
Qwen2.5-3B	75.00	**90.91**	68.97	94.57	96.18	84.65
Qwen2.5-7B	**76.25**	86.36	**72.41**	**96.31**	**96.96**	**92.33**

Table 3: Results on PFDial and PFDial-H: Backward Acc represents the accuracy of backward transition.

refer to cases where there are at least two possible next states for the current state. Accordingly, our accuracy can be further refined into two types: sequential accuracy and decision accuracy.

3.2 Main Results

Table 2 presents our detailed experimental results on the main experiments. Our experiments demonstrate significant improvements over baseline models across all parameter scales. Even our smallest 0.5B parameter model achieves 91.35% accuracy on out-of-domain tests, with particularly strong performance on decision branches. Our 8B model achieves sota performance with 97.29% accuracy on out-of-domain tests, surpassing GPT-4o by 11%. In decision branches, our 8B parameter model achieves 96.88% accuracy, surpassing GPT-4o by 43.88%. See Appendix E for a detailed case study analyzing the reasons behind model errors.

3.3 Data Scaling Experiments

Experimental Setup To investigate data efficiency, we conducted experiments with varying training data sizes from 100 to 12,705 samples. We employed two data scaling strategies: **(a)** keeping fixed 12,000 general dialogue data samples while gradually increasing PFDial data, and **(b)** maintaining a 1:1 ratio mixing of general dialogue data and PFDial data.

Results and Analysis Figure 3 presents our experimental results on the data scaling experiments. The results demonstrate remarkable performance even with limited data: using only 800 samples, a 7B model can surpass 90% accuracy on OOD test. Performance continues to improve consistently with increased data volume, though we observe diminishing returns after approximately 3,000 samples. The comparable performance across both data scaling strategies validates the effectiveness of our PFDial dataset, demonstrating its robust data efficiency regardless of the mixing approach. For comprehensive results on decision accuracy and sequential accuracy across various data scaling configurations, please refer to Appendix C.2.

3.4 Backward Transition Studies

Experimental Setup We evaluated models' capability in handling complex backward transitions on the PFDial-H test set, which provides a more rigorous assessment of models' ability to strictly follow process constraints. The details of PFDial-H test set is shown in table 5 in Appendix B.2. For handling backward transitions, we compared two approaches: integrated training, which incorporates 440 PFDial-H training data samples to our PFDial training data during initial training, and secondary fine-tuning, which applies a secondary fine-tuning phase using PFDial-H data on the SFT-completed model with 440 PFDial-H training data samples.

Results and Analysis Detailed experimental results are shown in Table 3. Our results demonstrate that our method achieves optimal performance in handling backward branches, with the integrated training approach yielding superior results by main-

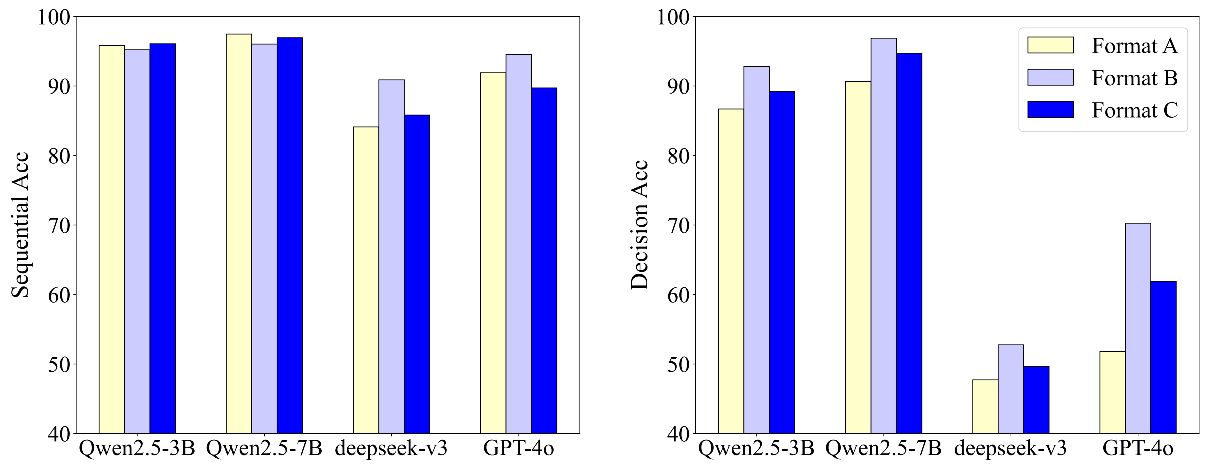


Figure 4: Results on Model Performance with Different Formats: The left plot shows the sequential accuracy for different models across three different formats (Format-NL, Format-SC, and Format-Hybrid). The right plot shows the decision accuracy for the same models under the same formats.

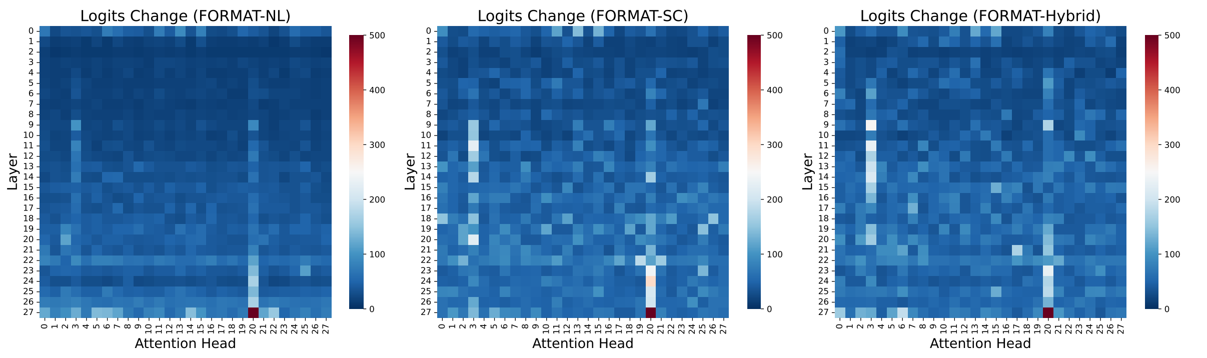


Figure 5: Logits changes of Three formats

taining robust performance on forward transitions while significantly improving accuracy in backward transition cases. Specifically, with integrated training, the Qwen2.5-7B model achieves 76.25% overall accuracy on PFDial-H, with 86.36% accuracy on backward transitions with distance less than 5, and 72.41% accuracy on transitions with distance greater than or equal to 5. Meanwhile, the model maintains a high accuracy of 96.31% on the OOD test.

In contrast, secondary fine-tuning not only fails to improve performance on backward transition cases but also reduces performance on the PFDial dataset, with Qwen2.5-7B's OOD test accuracy dropping from 96.31% to 76.04%. These results emphasize the importance of integrating backward transition samples during the initial training process rather than treating them as a post-hoc fine-tuning step.

3.5 Format Abliation Studies

Experimental Setup We conducted experiments for three state representation formats: Format-NL: natural language description (default method), Format-SC: state codes (e.g., S1, C2), and Format-Hybrid: combining codes and descriptions to explore the impact of different data formats on model performance and the underlying reasons. Specific cases of data in different formats and the visualization results can be seen in Appendix C.3. To ensure fairness, we fine-tuned the base model on data in all three formats and tested it with corresponding test sets containing the same content.

We then froze the attention heads of each layer and compared the model's output logits with the original logits in these scenarios. By examining the logits changes , we assessed the impact of each attention head after fine-tuning with different formats. Finally, we visualized these attention heads to gain deeper insights into their roles.

Results and Analysis The model performance with different formats are shown in Figure 4. The results indicate that Format B achieved the highest accuracy in most cases, particularly on Decision Accuracy, however, it performed slightly worse than other formats on Sequential Accuracy. The following experimental results, to some extent, shed light on the reasons behind this phenomenon.

The comparison results of the three formats's attention heads are shown in Figure 5. The models fine-tuned with FormatB and FormatC showed a more uniform and diverse distribution of attention head contributions. This can be explained that both of the latter formats introduced code state identifiers, requiring the language model to learn both the sequence of state transitions and the correspondence between code states and natural language states. Thus more attention heads with different functions were activated.

In particular, $head\_layer_{27}\_head_{20}$ is crucial in all three formats. We visualized this attention head's scores for all three formats in Figure 6, Figure 7, and Figure 8, respectively, focusing on significant differences. The attention scores for both the natural language and hybrid formats were concentrated at the intersection of the user's current state and the corresponding state in the flowchart. Format-Hybrid, which includes some state codes, showed a more moderated concentration. In contrast, Format-SC did not exhibit such clustering. We hypothesize that the introduction of state codes allows the model's attention to generalize across different input parts, rather than being confined to specific segments. This enables models to better understand user inputs and facilitates learning of global logic, such as condition checking and state selection. This also reasonably explains previous results.

4 Related Work

4.1 Controllable Reasoning in LLMs

Controllable reasoning in LLMs has gained significant attention in recent years. Ouyang et al. (2022b) pioneered instruction-guided control via Reinforcement Learning from Human Feedback (RLHF), combining supervised fine-tuning (SFT), reward model training, and Proximal Policy Optimization (PPO) to align outputs with human preferences. While effective, this approach requires complex annotation and training (Li and Liang, 2021). In contrast, our approach simplifies the process by encoding reasoning logic into structured UML state flowcharts, guiding learning through SFT alone. This provides a clear, human-readable control mechanism, addressing the 'reasoning opacity' challenge (Liang et al., 2024b).

4.2 Graph-based Enhanced Reasoning

Previous research (Pan et al., 2024) has explored graph-based approaches to enhance the reasoning capabilities of LLMs. Several works (Liang et al., 2024a; Luo et al., 2024a,b) have used Knowledge Graph (KG) structural information to reduce hallucinations by breaking reasoning into path extraction and inference steps. Similarly, Zhou et al. (2024) showed that graph-based training improves multi-hop reasoning accuracy. However, these methods mainly treat graphs as external knowledge sources rather than explicit control mechanisms.

4.3 LLM-based Dialogue Systems

Task-Oriented Dialogue (TOD) systems help users achieve specific goals through conversations (Yi et al., 2024). Current approaches fall into two main categories: Pipeline-based Approaches, which separate dialogue systems into modules with LLMs handling specific tasks (Comi et al., 2023; Parikh et al., 2023; Chen et al., 2019; Nguyen et al., 2023), and End-to-End Approaches, which use LLMs to generate responses based on the entire dialogue history (Hemanthage et al., 2023; Zhang et al., 2024, 2023; Wu et al., 2020; Algherairy and Ahmed, 2025). Pipeline-based Approaches offer better transparency but require extensive annotated data, while End-to-End Approaches are simpler but less controllable and demand higher model capabilities.

5 Conclusion

In this paper, we introduce the PFDial dataset, a novel resource designed to enhance process-driven dialogue systems. By utilizing structured dialogue instructions derived from UML flowcharts, PFDial provides a robust framework for training models to handle complex decision-making and sequential processes. Our experiments demonstrate that models fine-tuned on PFDial achieve high accuracy, even with limited training data, and outperform sota LLMs like GPT-4o on specific tasks.

We conducted an in-depth analysis of backward transitions using the PFDial-H dataset, highlighting the importance of integrated training approaches for maintaining strong performance across diverse dialogue scenarios. Additionally, we explored the impact of various data representation formats, finding that structured state codes significantly improve the accuracy of state transition predictions.

Overall, our work underscores the potential of structured datasets like PFDial to advance process-driven dialogue systems, offering new insights into the design and training of models for precise and controlled reasoning. Future research will focus on expanding the dataset to cover more scenarios and refining training methodologies to enhance model generalization and adaptability.

Limitations

Our research presents a comprehensive set of experiments, yet it is not without limitations. First, the Chinese-centric nature of our dataset introduces potential cross-lingual generalization constraints. Second, the scarcity of standardized flowchart benchmarks in Chinese process specifications increases the risk of domain-specific biases, despite our rigorous validation framework. Besides, the potential residual inconsistencies in flow-to-text conversion may emerge from the inherent subjectivity in interpreting semantic structures.

Acknowledgments

The authors wish to thank anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62376061,62206057), Shanghai Rising-Star Program (23QA1400200), Natural Science Foundation of Shanghai (23ZR1403500), Program of Shanghai Academic Research Leader under grant 22XD1401100.

References

Atheer Algherairy and Moataz Ahmed. 2025. Prompting large language models for user simulation in task-oriented dialogue systems. *Comput. Speech Lang.*, 89:101697.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

BELLEGroup. 2023. Belle: Be everyone's large language model engine. `https://github.com/LianjiaTech/BELLE`.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

Daniele Comi, Dimitrios Christofidellis, Pier Francesco Piazza, and Matteo Manica. 2023. Zero-shot-bert-adapters: a zero-shot pipeline for unknown intent detection. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 650–663. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, and Wangding Zeng. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Bhathiya Hemanthage, Christian Dondrup, Phil Bartie, and Oliver Lemon. 2023. Simplemtod: A simple language model for multimodal task-oriented dialogue with symbolic scene representation. *CoRR*, abs/2307.04907.

Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. 2024. Openrlhf: An easy-to-use, scalable and high-performance RLHF framework. *CoRR*, abs/2405.11143.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Xun Liang, Hanyu Wang, Shichao Song, Mengting Hu, Xunzhi Wang, Zhiyu Li, Feiyu Xiong, and Bo Tang. 2024a. Controlled text generation for large language model with dynamic attribute graphs. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 5797–5814. Association for Computational Linguistics.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024b. Controllable text generation for large language models: A survey. *CoRR*, abs/2408.12599.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024a. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 2039–2056. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hoang Nguyen, Ye Liu, Chenwei Zhang, Tao Zhang, and Philip S. Yu. 2023. Cof-cot: Enhancing large language models with coarse-to-fine chain-of-thought prompting for multi-domain NLU tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12109–12119. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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. 2022a. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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. 2022b. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Trans. Knowl. Data Eng.*, 36(7):3580–3599.

Soham Parikh, Mitul Tiwari, Prashil Tumbade, and Quaizar Vohra. 2023. Exploring zero and few-shot techniques for intent classification. In *Proceedings of the The 61st Annual Meeting of the Association for Computational Linguistics: Industry Track, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 744–751. Association for Computational Linguistics.

Cheng Wen, Xianghui Sun, Shuaijiang Zhao, Xiaoquan Fang, Liangyu Chen, and Wei Zou. 2023. Chathome:

Development and evaluation of a domain-specific language model for home renovation. *arXiv preprint arXiv:2307.15290*.

Chien-Sheng Wu, Steven C. H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 917–929. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *CoRR*, abs/2402.18013.

Ming Zhang, Caishuang Huang, Yilong Wu, Shichun Liu, Huiyuan Zheng, Yurui Dong, Yujiong Shen, Shihan Dou, Jun Zhao, Junjie Ye, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Transfertod: A generalizable chinese multi-domain task-oriented dialogue system with transfer capabilities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 12750–12771. Association for Computational Linguistics.

Xiaoying Zhang, Baolin Peng, Kun Li, Jingyan Zhou, and Helen Meng. 2023. SGP-TOD: building task bots effortlessly via schema-guided LLM prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 13348–13369. Association for Computational Linguistics.

Jiaming Zhou, Abbas Ghaddar, Ge Zhang, Liheng Ma, Yaochen Hu, Soumyasundar Pal, Mark Coates, Bin Wang, Yingxue Zhang, and Jianye Hao. 2024. Enhancing logical reasoning in large language models through graph-based synthetic data. *CoRR*, abs/2409.12437.

A Algorithm

A.1 Parse PlantUML Code

This algorithm parses the given PlantUML code by line by line. It initializes an empty dictionary, *nodeDict*, to store state nodes. Then, it calls PARSESEQUENTIAL (Algorithm 2) to process the sequential flow. Finally, it returns all paths originating from the start node, representing the complete execution flow.

Algorithm 1 Parse PlantUML Code

```
1: function PARSEPLANTUML(code)
2:     Split code into lines
3:     Initialize an empty dictionary nodeDict to store nodes
4:     Call PARSESEQUENTIAL
5:     return all paths originating from the start node
6: end function
```

A.2 Parse Sequential Blocks

This algorithm parses the sequential execution flow by processing each line to identify states. Sequential states create and merge new nodes, while decision states call PARSEDECISION (Algorithm 3) for further processing.

Algorithm 2 Parsing Sequential Blocks

```
1: function PARSESEQUENTIAL(startNode, lines, nodeDict)
2:     root ← startNode
3:     for each line in lines do
4:         Trim whitespace
5:         if the line represents a sequential state then
6:             Create a new node into nodeDict
7:             Merge new nodes
8:         else if the line represents a decision state then
9:             Call PARSEDECISION
10:            Connect new nodes
11:        else
12:            Continue processing
13:        end if
14:    end for
15:    return
16: end function
```

A.3 Parse Decision Blocks

This algorithm handles decision structures by first parsing the "if" block using PARSESEQUENTIAL (Algorithm 2). If an "else if" block is encountered, it recursively calls itself to process nested conditions. For an "else" block, it parses the sequence and connects the resulting nodes using PARSESEQUENTIAL (Algorithm 2). This ensures correct branching and flow control within decision blocks.

Algorithm 3 Parsing Decision Blocks

```
function PARSEDECISION(startNode, lines, nodeDict)
    root ← startNode
    Call PARSESEQUENTIALto parse “if” block
    if meet “else if” block then
        Recursively call PARSEDECISION
        Connect new nodes
        return
    else if meet “else” block then
        Parse the “else” block using PARSESEQUENTIAL
        Connect new nodes
        return
    else
        return
    end if
end function
```

B Dataset

B.1 Different Format to Represent Flowchart

In Table 4, we compares different formats to represent flowchart. We select a complex flowchart with 21 paths, and let GPT-4o to find all paths and count the number paths. Only with PlantUML, GPT-4o correctly output the right path number, which means flowchart with PlantUML is easy for models to understand. Moreover, syntax features, as is shown in Table 4, such as indentation branching, and loops, making PlantUML a good format to read and convenient for program processing.

B.2 PFDial-H Tests

Table 5 presents the details of PFDial-H tests, including the total number of dialogues in the test set, average dialogue length in turns, and the proportion of dialogues with backward transition distances divided by the threshold of 5.

C Experimental Details

C.1 Implementation Details

To ensure reproducibility of our experiments, we provide detailed hyper-parameter configurations used in our training process. All experiments were conducted using the AdamW optimizer with cosine annealing learning rate scheduling. The complete set of training hyper-parameters is presented in Table 6. We maintained consistent hyper-parameter settings across models of different scales to ensure fair comparison.

C.2 Supplementary Data for Data Scaling Experiment

While the section 3.3 presents the overall accuracy trend with increasing training samples, here we provide the complete experimental results. Tables 13 and 14 present the performance results for Strategy A and Strategy B respectively. Each table shows overall accuracy, decision accuracy, and sequential accuracy metrics on both ID and OOD tests across different training sample sizes. The information of dataset with different training sample sizes is shown in Table 12.

C.3 Details in Format Ablation Study

In this section, we present specific cases of data in different formats and the visualization results for $head\_layer_{27}\_head_{20}$. As shown in Figure 6, 7, and 8, we visualize the attention patterns for Format-NL (Natural Language) in Table 15, Format-SC (State Code) in Table 16, and Format-Hybird in Table 17 respectively.

D Prompt

D.1 Prompt for generating User Input

We use the prompt from Table 7 to generate user inputs. This prompt helps us create appropriate user input text based on the given state transitions.

D.2 Prompt for generating Robot Output

We use the prompt from Table 8 to generate robot outputs. This prompt helps us create appropriate robot responses based on the current state, next state, and user input.

Format	PlantUML	ChartMage	NomNoml	Meamaid

Decision Block	if (D) then (C1)
	S1
	else (C2)
	S2
		D - C1 ->> S1
		D - C2 ->> S2
			[D] C1-> [Block1]
			[D] C2-> [Block2]
				A{D} – C1 –> B[S1]
				A – C2 –> C[S2]

Input	Here is the flowchart code: [a UML flowchart with 21 paths, expressed in the corresponding format] Show all possible complete paths and count how many there are.			
Path Count	21	15	19	17

Table 4: Comparison of Flowchart Formats

Backward Distance < 5	22
Backward Distance $\geq$ 5	58
Dialogue Samples	80
Avg. Length	534.36

Table 5: PFDial-H Tests Data Overview

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	5×10^{-6}
Learning Rate Scheduling	Cosine Annealing
Adam Beta1	0.9
Adam Beta2	0.95
Batch Size	128
Batch Size Per-Device	4
Training Epochs	5

Table 6: Training Hyper Parameters

User

These examples are five-tuples consisting of the PlantUML diagram, the current state, the next state, user input, and the robot output.

[several examples]

The user's input explains the change in state from the current state to the next state. The robot output is related to next state. Robot acts as the server-provider. For example, if the current state is A, tobot might output "Now process A." or, when a choice is required, robot lets user to make a choice. Now I have a five-tuple consisting of the PlantUML diagram, the current state next state, and the user input, without robot output. Your task is to generate the robot's output based on the rules provided.

This is the five-tuple need to be filled:
[five-tuple to be filled]

Table 8: The prompt for generating the robot's output.

User

These examples are four-tuples consisting of the PlantUML diagram, the current state, the next state, and the user input.

[several examples]

The user's input explains the change in state from the current state to the next state. For example, if the original state is A, the user might input "A has been completed." or, when a choice is required, the user selects an option based on the next state.
Now I have a four-tuple consisting of the PlantUML diagram, the current state, and the next state, without user inpupt. Your task is to generate the user's input based on the rules provided.

This is the four-tuple need to be filled:
[four-tuple to be filled]

Table 7: The prompt for generating the user's input.

D.3 Prompt for Adding Backward Transition

In the third step, we use the prompt from Table 9 to add backward transitions. This prompt guides us in adding logical loop structures to the flowchart.

User
This is a flowchart in PlantUML syntax and the result after adding a loop to itself: **[original PlantUML and revised PlantUML]** Your task is to follow this modification rule ro add a loop to the plantuml that I will give you next. The following conditions must be met: 1. The added loop is logical 2. The conditional state of "repeat while" cannot be repeated with the any conditional state that already exists in the original PlantUML 3. Ensure that the syntax of PlantUML is correct 4. Add is([need to loop]) not([jump out of loop]) statements after repeat while as much as possible. PlantUML to be modified: **[original PlantUML]**

Table 9: The prompt for Adding Backward Transition.

E Case Study

We analyzed specific cases to provide intuitive explanations in Table 10 and 11. First, generally speaking, these models fail because they don't precisely provide complete descriptions of the next state, or they generate states related to user input but not in the original flowchart, or they don't follow the format required in the system prompt. Our task examines models' ability to predict the next state in sequential branches and decision branches. Below we provide examples and analyze why GPT-4o performs poorly in these areas.

E.1 Sequential Branch

For sequential branches, the model only need to capture the current state and state transition. This is typically simple because adjacent states in PlantUML are usually adjacent lines. However, when multiple decision branches are nested, things become complicated and the line distance between adjacent states increases. For example, in Table 10, after "Repeatedly check if printing is complete", all nested loops have ended and should transition to "Customer leaves the photo shop". However, due to the high complexity caused by multiple nested sequential branches, GPT-4o struggles to accurately capture the contextual state relationships, resulting in errors.

E.2 Decision Branch

For decision branches, the model needs to precisely transition to the next state based on the current state and user input. Unfortunately, in most wrong cases, due to the high similarity between user input and the first state of the correct branch during condition judgment, the model tends to skip this first state and directly enter subsequent states, failing to accurately follow the agreed path. More challenging issues arise when decision branch conditions involve more than just yes/no decisions, requiring models to comprehensively understand user input and judge their true intentions. For example, in Table 11, when the user inputs "The position of the lighting fixture needs to be adjusted.", it indicates that the lighting position needs to be modified at this point. However, the model struggles to strictly adhere to the process constraints and makes an incorrect state prediction based on the user input, skipping the correct state of this branch.

PlantUML

```
@startuml
start
:Customer arrives at photo shop;
:Submit photo files for printing;
:Select print size and quantity;
:Choose paper quality and surface effect;
:Confirm print order and price;
if (Photo quality check passed?) then (yes)
    :Pay fee;
    :Photo printing in progress;
    if (Printing completed?) then (yes)
        :Display printed photos;
        :Customer satisfied?;
        if (Customer satisfied?) then (yes)
            :Complete transaction;
        else (no)
            :Reselect photos;
            :Reprint;
            :Complete transaction;
        endif
    else (no)
        :Wait for photo printing;
        :Repeatedly check if printing is complete;
    endif
else (no)
    :Notify poor photo quality;
    :Reselect photos;
    :Reprint;
    :Complete transaction;
endif
:Customer leaves the photo shop;
stop
@enduml
```

Current State	Repeatedly check if printing is complete
User Input	Printing is completed.
Predicted Next State	`Display printed photos` (Incorrect)
Correct Next State	`Customer leaves the photo shop` (Correct)

Table 10: Case Study in Sequential Branch

PlantUML	`@startuml`
	`start`
	`:Communicate lighting design plan with construction team;`
	`:Explain design requirements and fixture layout;`
	`:Provide technical support and guidance;`
	`:Coordinate installation time and progress;`
	`:Supervise fixture installation process;`
	`:Perform fixture debugging and brightness adjustment;`
	`repeat`
	`:Confirm fixture layout and installation position;`
	`:Supervise fixture installation process;`
	`repeat while(Installation and debugging unsuccessful)`
	`:Check fixture installation quality and safety;`
	`if (Need to adjust fixture position?) then (yes)`
	`    :Negotiate adjustment plan;`
	`    :Reinstall or adjust fixture position;`
	`else (no)`
	`    :Confirm final installation result;`
	`endif`
	`:Final acceptance of lighting system;`
	`:Ensure lighting system meets design requirements;`
	`:Complete construction documents and records;`
	`stop`
	`@enduml`

Current State	Need to adjust fixture position?
User Input	The position of the lighting fixture needs to be adjusted.
Predicted Next State	`Reinstall or adjust fixture position` (Incorrect)
Correct Next State	`Negotiate adjustment plan` (Correct)

Table 11: Case Study in Decision Branch

Training Sample Size	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**9600**	**12705**
Flowcharts	3	5	12	29	59	113	208	308	440
State Nodes	49	79	149	304	612	1236	2473	3772	5055
Sequential Samples	62	134	264	561	1099	2185	4293	6717	9029
Decision Samples	38	66	136	239	501	1015	2107	2883	3676
Avg. Length	386.22	368.19	320.24	286.92	275.63	272.29	273.00	280.90	277.16

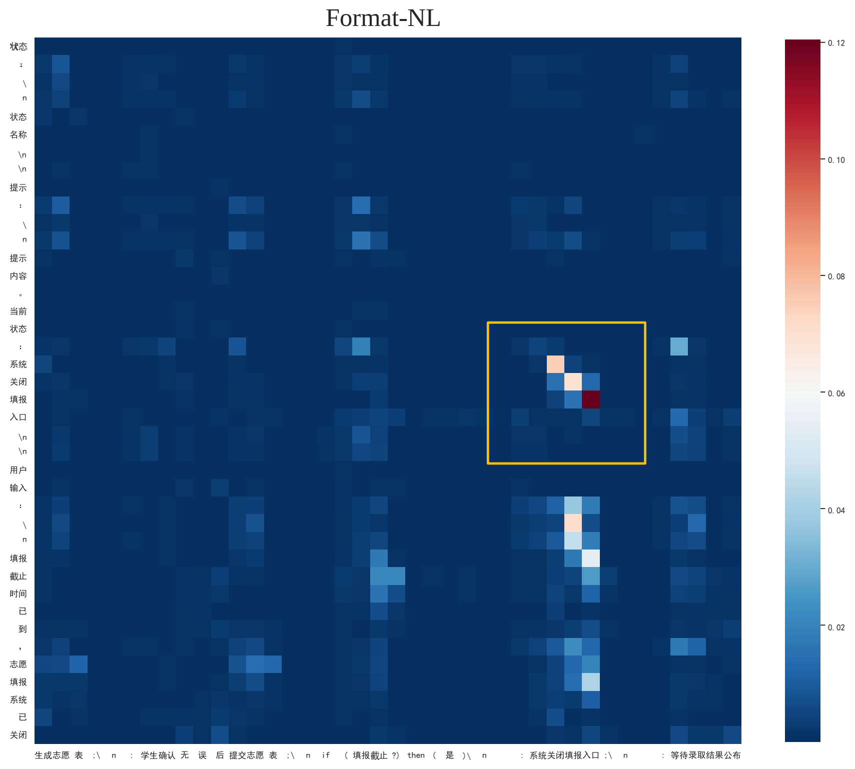
Table 12: Statistics of the PFDial Dataset with Different Training Sample Sizes.

Training Sample Size	**ID-test**			**OOD-test**		
	Acc	**Decision Acc**	**Sequential Acc**	**Acc**	**Decision Acc**	**Sequential Acc**
100-all	77.61	54.52	81.87	73.59	39.81	79.08
200-all	84.82	83.33	85.10	80.83	62.35	83.83
400-all	88.91	90.68	88.59	87.80	76.50	89.64
800-all	90.37	92.37	89.99	88.44	78.90	89.99
1600-all	93.84	94.92	93.64	91.59	84.17	92.79
3200-all	96.61	97.46	96.46	93.26	89.69	93.84
6400-all	97.89	97.18	98.02	95.34	91.85	95.91
9600-all	98.86	**98.31**	98.96	96.45	90.17	**97.47**
all	**98.94**	98.02	**99.11**	**96.51**	**90.65**	**97.47**

Table 13: Performance with different training sample sizes across ID and OOD datasets after training on Qwen2.5-7B, with data scaling strategy **(a)**.

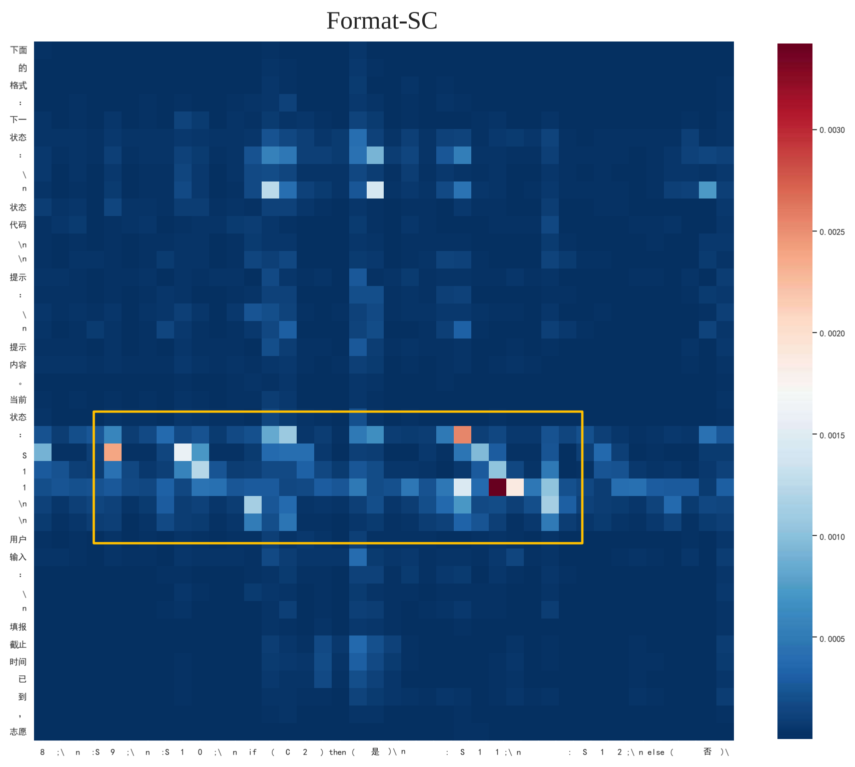
Training Sample Size	**ID-test**			**OOD-test**		
	Acc	**Decision Acc**	**Sequential Acc**	**Acc**	**Decision Acc**	**Sequential Acc**
100-100	61.37	23.45	68.37	59.12	13.91	66.46
200-200	82.31	84.46	81.92	77.88	65.47	79.90
400-400	89.71	89.55	89.73	86.03	75.30	87.77
800-800	91.55	92.66	91.35	90.01	81.06	91.47
1600-1600	93.80	93.50	93.85	91.79	85.13	92.87
3200-3200	97.01	98.02	96.82	94.30	90.65	94.90
6400-6400	97.98	98.31	97.92	95.54	90.65	96.34
9600-9600	98.90	97.46	**99.17**	95.95	89.45	97.00
all	**98.94**	**98.02**	99.11	**96.51**	**90.65**	**97.47**

Table 14: Performance with different training sample sizes across ID and OOD datasets after training on Qwen2.5-7B, with data scaling strategy **(b)**.



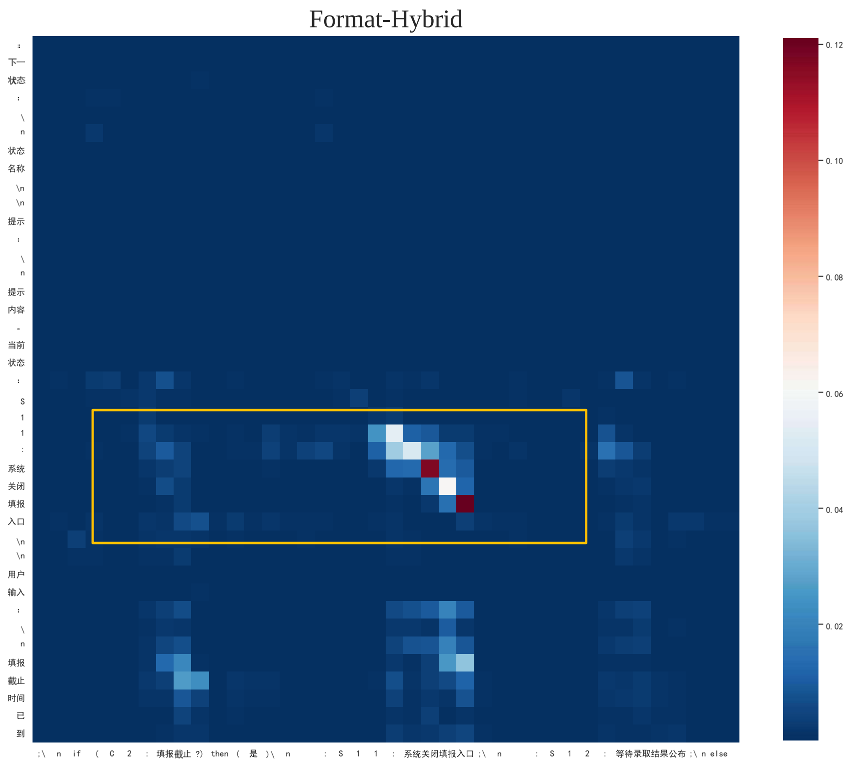
(a) Format-NL

Figure 6: local attention score of $head\_layer_{27}\_head_{20}$ using Format-NL



(b) Format-SC

Figure 7: local attention score of $head\_layer_{27}\_head_{20}$ using Format-SC



(c) Format-Hybrid

Figure 8: local attention score of $head\_layer_{27}\_head_{20}$ using Format-Hybrid

Format	Data Example
Format-NL	**PlantUML:** @startuml start :College entrance exam results announced; :Student obtains college entrance exam score report; :Student checks the exam results and determines the range of colleges and majors to apply for; if (Do you need to research colleges and majors in advance?) then (Yes) :Student conducts research on colleges and majors; else (No) :Skip this step; endif :Student logs into the application system; :System provides the application entry and instructions; :Student fills out the application, prioritizing choices; :After completing the application, the system generates an application form; :Student confirms the form and submits it; if (Application deadline?) then (Yes) :System closes the application entry; :Wait for the admission results to be announced; else (No) :Student can modify the application before the deadline; :Wait for the application deadline; endif :Admission results announced; if (Admitted?) then (Yes) :Student completes registration procedures according to the admission notice; else (No) :Student applies for supplementary applications or participates in the supplementary application process; endif :Enroll in the school; stop @enduml **Current state:** System closes the application entry **Next state:** Wait for the admission results to be announced **User input:** The application deadline has passed, and the application system is now closed. **Robot output:** Please patiently wait for the announcement of the admission results.

Table 15: Example Data Display - Format-NL

Format	Data Example
Format-SC	**PlantUML:** @startuml start :S1; :S2; :S3; if (C1) then (Yes) :S4; else (No) :S5; endif :S6; :S7; :S8; :S9; :S10; if (C2) then (Yes) :S11; :S12; else (No) :S13; :S14; endif :S15; if (C3) then (Yes) :S16; else (No) :S17; endif :S18; stop @enduml **Dictionary of the state codes:** { "College entrance exam results announced": "S1", "Student obtains college entrance exam score report": "S2", "Student checks the exam results and determines the range of colleges and majors to apply for": "S3", "Do you need to research colleges and majors in advance?": "C1", "Student conducts research on colleges and majors": "S4", "Skip this step": "S5", "Student logs into the application system": "S6", "System provides the application entry and instructions": "S7", "Student fills out the application, prioritizing choices": "S8", "After completing the application, the system generates an application form": "S9", "Student confirms the form and submits it": "S10", "Application deadline?": "C2", "System closes the application entry": "S11", "Wait for the admission results to be announced": "S12", "Student can modify the application before the deadline": "S13", "Wait for the application deadline": "S14", "Admission results announced": "S15", "Admitted?": "C3", "Student completes registration procedures according to the admission notice": "S16", "Student applies for supplementary applications or participates in the supplementary application process": "S17", "Enroll in the school": "S18" } **Current state:** System closes the application entry **Next state:** Wait for the admission results to be announced **User input:** The application deadline has passed, and the application system is now closed. **Robot output:** Please patiently wait for the announcement of the admission results.

Table 16: Example Data Display - Format-SC

Format	Data Example
Format-Hybrid	**PlantUML:** @startuml start :S1: College entrance exam results announced; :S2: Student obtains college entrance exam score report; :S3: Student checks the exam results and determines the range of colleges and majors to apply for; if (C1: Do you need to research colleges and majors in advance?) then (Yes) :S4: Student conducts research on colleges and majors; else (No) :S5: Skip this step; endif :S6: Student logs into the application system; :S7: System provides the application entry and instructions; :S8: Student fills out the application, prioritizing choices; :S9: After completing the application, the system generates an application form; :S10: Student confirms the form and submits it; if (C2: Application deadline?) then (Yes) :S11: System closes the application entry; :S12: Wait for the admission results to be announced; else (No) :S13: Student can modify the application before the deadline; :S14: Wait for the application deadline; endif :S15: Admission results announced; if (C3: Admitted?) then (Yes) :S16: Student completes registration procedures according to the admission notice; else (No) :S17: Student applies for supplementary applications or participates in the supplementary application process; endif :S18: Enroll in the school; stop @enduml **Current state:** System closes the application entry **Next state:** Wait for the admission results to be announced **User input:** The application deadline has passed, and the application system is now closed. **Robot output:** Please patiently wait for the announcement of the admission results.

Table 17: Example Data Display - Format-Hybrid

Category	**Senarios**
Lifestyle Services	Hairdressing, Phone Card, Car Wash, Agritainment, Printing & Copying, Lawyer, Yoga Studio, Music Classroom, Internship
Daily Convenience	Takeout, Scenic Spots, Furniture Cleaning, Wedding Photography, Movie, Cleaning Service, Self-service Car Wash, Second-hand Car Trading, Visa Application
Food & Education	Catering, Training Courses, Physiotherapy & Massage, Lighting Design, Gym, Express Delivery, Travel Group Purchase, Health Check-up, Group Tour
Entertainment & Transportation	Concert, Car Rental, Baking Studio, Digital Repair, Airplane Ticket, Water Delivery, Florist Shop, Photo Studio, Course Selection
Business & Professional Services	Commercial Photography, Hospital, Pet Boarding, Café, Dentist, Pet Grooming, Tea House, Outdoor Expansion, Electrician Inspection
Luxury & Specialized Services	Beauty Salon, Museum, Horticultural Design, Car Maintenance, Cruise, Photography Studio Rental, Piano Tuning, Basketball Court, Cargo Delivery into Cabin
Living-related Services	Laundry, House Rental, Education Consultation, Library, Cultural Exhibition, Health Consultation, Holiday Villa, Interior Design, Bank Account Operation
Maintenance & Care Services	Home Appliance Repair, Hotel, Leather Goods Care, Arcade, Furniture Installation, Medical Check-up, Car Insurance Claim, Bicycle Rental, Parts Inspection
Comprehensive Services	Moving, Resort, Language Translation, Medical Aesthetics, Driving School, Wedding Planning, Pet Hospital, Manicure, Document Approval
Shopping & Other Activities	Shopping, Train Ticket, Water & Electricity Repair, Ski Resort, Credit Card, College Entrance Examination Volunteer Filling, Cooking, DIY Handicraft, Content Creation

Table 18: Senarios