Perceive Before Reasoning: A Pre-Reasoning Perception Framework for Efficient and Reliable Proactive Mobile Agents

Anonymous ACL submission

Abstract

Multimodal large language models (MLLMs) have substantially advanced mobile agents, yet proactive mobile assistance remains challenging because agents must decide When to intervene before determining How to assist. Existing systems often implement these two decisions within a unified MLLM-based pipeline, causing goal misalignment between conservative intervention filtering and comprehensive assistance generation, as well as redundant inference when the agent should remain silent. To address these limitations, we propose the **Pre-Reasoning Perception Framework (PRPF)**, a two-stage framework built on perceiving before reasoning. PRPF introduces a lightweight Multimodal Proactive Perceptor (MPP) for intervention gating and context compression, and activates the Proactive Agent Reasoner (PAR) only when intervention is warranted. Experiments on the ProactiveMobile benchmark show that PRPF substantially reduces false trigger rates (FTR) while improving success rates (SR) and inference efficiency over the ProactiveMobile baseline. Our code and data are available at https://anonymous.4open.science/r/PRPF.

1 Introduction

Multimodal large language models (MLLMs), particularly vision-language models (VLMs), have substantially advanced mobile agents by enabling them to perceive mobile interfaces and execute user instructions (Bai et al., 2025; Hurst et al., 2024; Zhang et al., 2025b). However, mobile agents are increasingly expected to move beyond reactive instruction following toward proactive assistance, where they anticipate user needs and intervene without explicit prompts (Lu et al., 2025; Kong et al., 2026). Existing studies commonly formulate proactive assistance as a When–How problem, in which an agent must first determine when to intervene and then decide How to act (Yang et al., 2026b; Xie et al., 2026; Liu et al., 2026a).

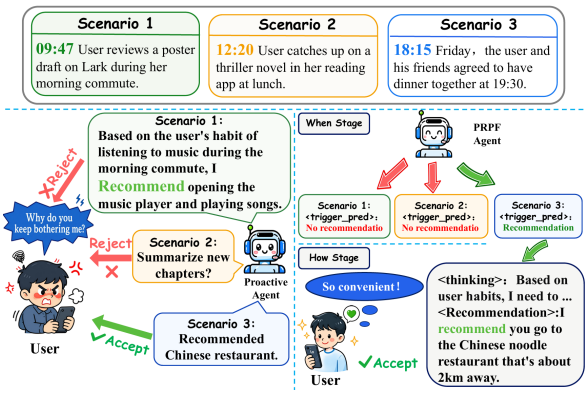


Figure 1: Comparison between unified proactive reasoning and PRPF.

Recent systems typically realize this formulation within a unified VLM-based agent. For example, ProAgentBench (Tang et al., 2026) treats when to assist and How to assist as timing judgment and content generation subtasks handled by the same fine-tuned backbone. Similarly, PARE-Bench (Nathani et al., 2026) frames proactive assistance as an Observe–Execute process, where observation and execution are realized by switching tool-use modes within one VLM-based assistant. Despite differences in implementation, these systems generally couple intervention timing and assistance generation within a single VLM-agent pipeline. This design gives rise to two key limitations:

Goal misalignment. The when decision requires conservative and highly discriminative judgment to suppress false triggers under asymmetric intervention costs, whereas the How decision requires broad multimodal reasoning and flexible content generation. Coupling these goals within a single VLM-agent makes it difficult to simultaneously achieve conservative intervention filtering and comprehensive assistance generation. As observed in Proactive Agent (Lu et al., 2025) and PRISM (Fu et al., 2026), even strong models strug-

gle to maintain the low false-trigger rates required for satisfactory mobile user experience. Excessive interventions disrupt user workflows, while overly conservative policies risk reducing proactive agents to passive assistants.

Inference inefficiency. Existing single-stage VLM agents execute the full reasoning pipeline regardless of whether intervention is ultimately needed. As a result, large models are activated even when the correct behavior is to remain silent, incurring unnecessary long-context and multi-turn reasoning costs (Gao et al., 2024; Yang et al., 2026a). This inefficiency becomes more severe in mobile environments with diverse scenarios and large function spaces. For example, ProactiveMobile (Kong et al., 2026) requires reasoning over schemas spanning 14 scenarios, substantially increasing prompt length and inference cost. ContextAgent (Yang et al., 2026b) similarly reports higher computational costs with limited performance gains under complex contextual settings.

To address these limitations, we propose the **P**re-**R**easoning **P**erception **F**ramework (**PRPF**), a mobile proactive intelligence framework built on the principle of perceiving before reasoning. Rather than using a large VLM to jointly determine when to intervene and How to assist, PRPF separates the two stages at the architectural level. A lightweight perceptual front-end first performs intervention gating and evidence compression, assigning the conservative when judgment to an efficient discriminative module. The VLM-based reasoner is then activated only when intervention is warranted, allowing it to focus on the generative How stage. This separation aligns each module with its corresponding goal while avoiding full VLM inference in cases where the agent should remain silent.

Concretely, PRPF introduces a **Multimodal Proactive Perceptor (MPP)**, a lightweight multimodal fusion encoder with short- and long-term pathways that operates before large-model reasoning. MPP first decides whether the agent should intervene or remain silent, thereby reducing false triggers at the source. It then compresses the multimodal context into a small set of Top-K scenario candidates, filtering irrelevant long-tail contexts and reducing the reasoning burden for the subsequent stage. Given this compact evidence, the **Proactive Agent Reasoner (PAR)** conducts focused deep reasoning to generate the final proactive recommendation. As a result, MPP serves as a pre-reasoning perceptual cutoff, while PAR concentrates its reasoning capacity on cases where proactive assistance is actually needed. Experiments on the ProactiveMobile benchmark show that PRPF improves the success rate from 20.82% to 41.15%, reduces the false trigger rate from 13.76% to 7.21%, and cuts expected inference compute by 69.3%. This demonstrates that PRPF, while substantially reducing inference compute, not only significantly enhances the accuracy of proactive recommendations and tool invocations, but also reduces false interruptions to users, knowing when to remain silent. In summary, our contributions are as follows:

- We propose PRPF, a two-stage pre-reasoning perception framework that architecturally decouples when judgment from How reasoning for proactive mobile agents.
- We design MPP, a lightweight and plug-and-play multimodal Perceptor for intervention gating and context compression, together with PAR, a Proactive Agent Reasoner for complete and focused How-stage reasoning.
- Extensive experiments on the ProactiveMobile benchmark demonstrate that PRPF substantially reduces false trigger rates while improving success rates and inference efficiency.

2 Related Work

2.1 Proactive Agents

Proactive-agent research extends LLM agents from instruction following to active assistance, where agents must judge intervention timing and generate useful assistance from ongoing context (Lu et al., 2025; Tang et al., 2026; Fu et al., 2026). Prior work studies this distinction through intervention-timing prediction, assistance-content generation, cost-sensitive selective intervention, and staged proactive behavior (Tang et al., 2026; Fu et al., 2026; Nathani et al., 2026; Xie et al., 2026). Proactive GUI and mobile settings make this problem concrete because user intent must be inferred from interface trajectories before an explicit command is issued. Existing benchmarks study intent recommendation over GUI trajectories and executable function-sequence generation from on-device context (Chai et al., 2026; Kong et al., 2026). Using ProactiveMobile as the evaluation setting, PRPF focuses on the architectural separation between lightweight pre-reasoning intervention perception

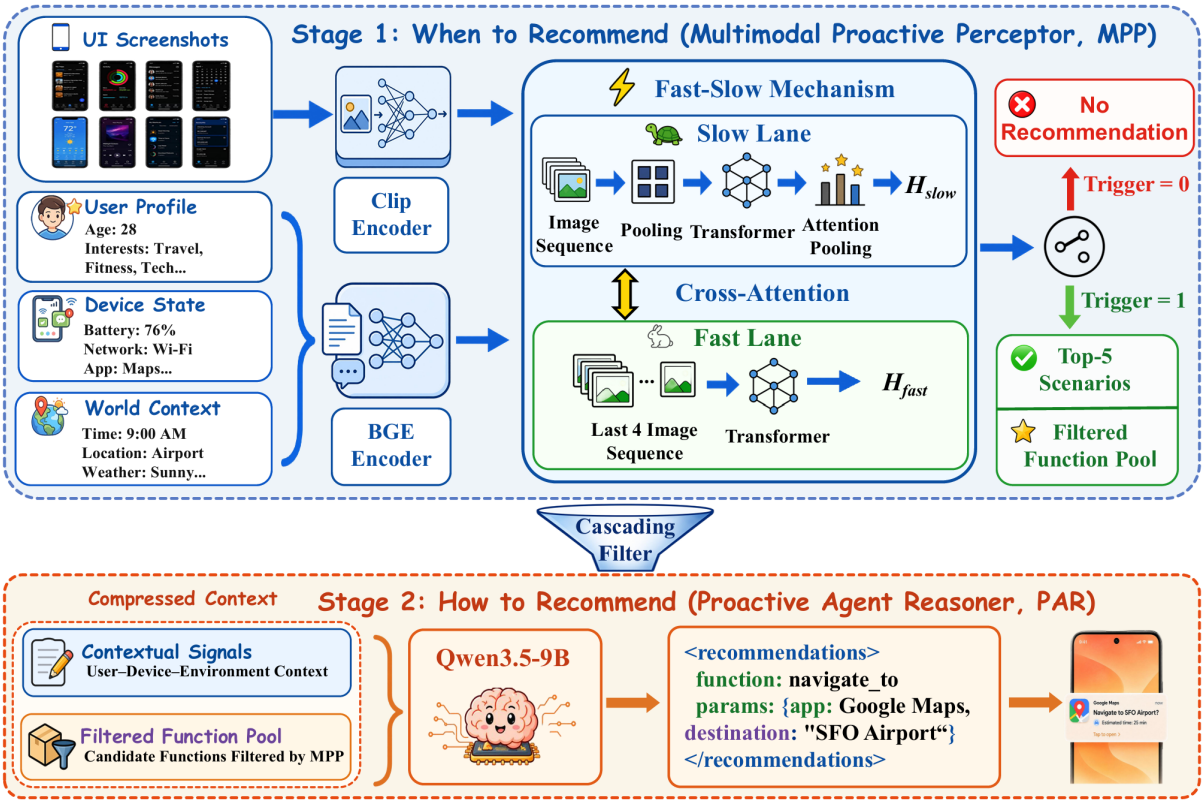


Figure 2: Overall framework of PRPF.

and heavy VLM-based assistance reasoning for accepted contexts.

2.2 GUI Perception and Efficient Reasoning

Mobile and GUI agents provide the perception and execution substrate for proactive assistance, but most existing systems remain reactive. Prior work has advanced smartphone operation, mobile task execution, screenshot-based GUI understanding, and visual grounding under explicit instructions (Zhang et al., 2025b; Wang et al., 2024; Rawles et al., 2025; Deng et al., 2024; Qin et al., 2025; Cheng et al., 2024). These advances improve how an agent perceives and acts on a requested GUI task, but not whether a continuously observed mobile context warrants intervention. Efficiency-oriented work further introduces intermediate perception, adaptive visual processing, region selection, or model routing before expensive reasoning (Wu et al., 2025; Mehrotra et al., 2025; Tang et al., 2025; Liu et al., 2026b; Ong et al., 2024; Xu et al., 2025). These methods use front-end selection, but their decision signals are typically task difficulty, region relevance, or model-call cost for a given query. They do not by themselves resolve the proactive intervention variable: whether a mobile GUI context warrants intervention and which intent candidates should condition subsequent assistance.

3 Method

3.1 Task Definition

Proactive intelligence on mobile devices requires models to infer users' latent needs before explicit requests and generate executable function-call sequences (Kong et al., 2026). Therefore, this task not only requires accurately determining when to trigger recommendations, but also generating function sequences aligned with user intent. Given a mobile interaction sample, its input includes the user profile (U), device state (D), world information (W), and interaction history (I). In the multimodal setting, I is represented as a GUI screenshot sequence, whereas in the text setting, it is represented as trajectory text. The model needs to produce an executable function-call sequence from the function set (F):

$$\hat{y} = \begin{cases} \varnothing \\ \hat{s} = [(\hat{f}_1, \hat{a}_1), \ldots, (\hat{f}_m, \hat{a}_m)] \end{cases} \quad (1)$$

where $\varnothing$ indicates that proactive recommendation should not be triggered under the current state; $\hat{s}$

denotes the function-call sequence predicted by the model; $\hat{f}_m \in F$ denotes the m-th predicted function, and $\hat{a}_m$ denotes the corresponding function argument.

3.2 PRPF Framework

To address the high false-trigger rate and inference inefficiency in proactive mobile intelligence, we propose the Pre-Reasoning Perception Framework (PRPF) , as shown in Figure 2. PRPF consists of two parts: 1) the lightweight Multimodal Proactive Perceptor (MPP), which quickly determines whether the current mobile interaction state should trigger proactive recommendation and predicts candidate functions; 2) the Proactive Agent Reasoner (PAR), which generates function-call sequences that are consistent with users' intent. For a data sample that is not recommended, MPP directly filters it out; otherwise, it enters PAR reasoning to generate the function-call sequence.

3.3 Multimodal Proactive Perceptor

MPP consists of a fast–slow dual-channel interaction module and two task-specific MLP heads for trigger gating and function-scenario prediction. The two heads determine whether to trigger proactive recommendation and predict candidate functions, providing upfront preprocessing for PAR.

Taking multimodal data as an example, the textual information and GUI screenshot sequence are first encoded by the text encoder and image encoder, respectively:

$$H_{\text{text}} = f_{\text{text}}(\{U, D, W\}), H_{\text{img}} = \{f_{\text{img}}(I_t)\}_{t=1}^{T}, \quad (2)$$

where f_{text} and f_{img} denote the text encoder and image encoder, respectively; I_t denotes the t-th GUI screenshot; and T denotes the length of the complete GUI screenshot sequence.
The textual and visual features are then projected into a unified dimensional space through linear mappings:

$$c_{text} = W_c H_{text}, \quad c_{img} = W_v H_{img}, \quad (3)$$

where W_c and W_v are the linear projection matrices. $c_{text} \in \mathbb{R}^{3\times d}$ and $c_{img} \in \mathbb{R}^{T\times d}$ denote the projected textual context representation and screenshot sequence representation, respectively, and d is the unified projection dimension.

The function-call sequence depends on both short-term GUI dynamics and long-term behavioral preferences reflected in the textual context. Accordingly, we construct fast and slow channels to capture these complementary signals.

The slow channel captures long-range behavioral patterns that reflect users' preferences. First, we apply attention pooling over the complete GUI sequence to obtain the global visual-state representation:

$$\bar{x} = \text{AttnPool}(H_{\text{img}}). \quad (4)$$

The pooled visual representation is then concatenated with c_{text} and fed into a Transformer Encoder to model the long-term cross-modal representation H_{slow} over U, D, W, and the complete GUI screenshot sequence.(Vaswani et al., 2017)

The fast channel captures short-term user dynamics. Based on empirical observations, we select the most recent four GUI screenshot representations and feed them through a projection layer and a Transformer Encoder to obtain the short-term dynamic representation H_{fast}.

$$H_{\text{fast}} = H_{\text{img}}[T-k+1:T], \quad k = 4. \quad (5)$$

To enhance the information interaction between the fast and slow channels, MPP introduces a cross-attention mechanism, enabling the model to simultaneously consider users' long-term behavioral background and the instantaneous state of the current interface when determining the recommendation timing. Specifically, for any channel pair $(\bar{c}, c) \in \{(\text{fast}, \text{slow}), (\text{slow}, \text{fast})\}$, the representation of the current channel is used as the Query, while the representation of the other channel is used as the Key and Value, thereby obtaining the cross-channel context-enhanced representation:

$$\widetilde{H}_c = \text{MHA}\left(Q = H_c, K = H_{\bar{c}}, V = H_{\bar{c}}\right), \quad (6)$$

then, attention pooling is applied to the two enhanced representations, which are concatenated to obtain the fused representation $z \in \mathbb{R}^{2d}$.

The fused representation z is fed into the trigger-gating head to obtain the probability of triggering proactive recommendation:

$$p_{trig} = \sigma(\text{MLP}_{trig}(z)), \quad (7)$$

where σ denotes the sigmoid function. If $p_{trig} < \tau$, the sample is directly filtered out. Otherwise, it is passed to PAR for subsequent reasoning. The details are provided in Appendix B.1.

To further compress the function candidate set and enhance reasoning efficiency, we feed z into

the function-scenario prediction head to obtain the probabilities of all scenarios:

$$p_{scenario} = \sigma(\mathrm{MLP}_{scenario}(z)). \quad (8)$$

We select the top-5 candidate function scenarios with the highest p_{scenario} and retain only the functions associated with these scenarios, thereby constructing the compressed candidate function set $\tilde{F}$. This filtering reduces the function space by approximately 68.25%. We present the quantitative analysis of MPP filtering effects in Appendix B.4.

3.4 Proactive Agent Reasoner

For samples judged as recommendation samples by MPP, we input U, D, W, the complete GUI screenshot sequence, and the compressed candidate function set $\tilde{F}$ into PAR for structured reasoning and content generation. Since MPP serves as a lightweight pre-reasoning gate, PAR performs a final consistency verification before generating executable recommendations, mitigating potential gating errors from the upstream perceiver. To better predict users' intent, we design a multi-step reasoning process.

First, UI summary is generated based on the GUI screenshots to understand the semantics and sequential information of the GUI screenshots. Then, combined with U, D, W, UI summary, and $\tilde{F}$, a thinking chain containing function selection is generated. Finally, PAR outputs the structured proactive recommendation results, including validated intervention decisions, selected function names, and corresponding parameters.

Reasoning Prompt $\mathcal{P}_r$

Given the user profile, device status, environmental information, screenshots, predicted scenes, and available functions, analyze the user's current behavior and determine whether a recommendation is needed.
1. ui_summary: Summarize the current interface, key content, and the user's operation stage.
2. thinking: Analyze the user's behavior, intent, task status, and whether a recommendation is needed; if needed, determine the recommendation function and parameters.
3. recommendations: Output the recommendation instruction and function call; if no recommendation is needed, output "No recommendation".

3.5 PRPF Training Objective

Since the two stages play different roles, we decouple PRPF into the training of two modules, MPP and PAR, and use the training set of ProactiveMobile (Kong et al., 2026) to train the two modules.

MPP is jointly optimized with two losses. Due to class imbalance, non-recommendation samples are oversampled to the same proportion as recommendation samples, and Focal Loss is used.(Lin et al., 2017) Given the ground-truth label y_{trig} and the predicted trigger probability p_{trig}, where $y_{\text{trig}} = 1$ indicates triggering recommendation, the trigger-gate loss is denoted as $\mathcal{L}_{\text{trig}}$:

$$p_t = y_{\text{trig}} p_{\text{trig}} + (1 - y_{\text{trig}})(1 - p_{\text{trig}}), \quad (9)$$

$$\mathcal{L}_{\text{trig}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \quad (10)$$

where p_t denotes the true-class predicted probability, γ reduces the weight of well-classified samples, and α_t balances different categories.

For triggered samples, BCE loss is used for optimization, and we compute the function-scenario prediction loss$\mathcal{L}_{scenario}$:

$$\mathcal{L}_{\text{scenario}} = \mathrm{BCE}\left(y_{\text{scenario}}, p_{\text{scenario}}\right), \quad (11)$$

where y_{scenario} denotes the corresponding ground-truth multi-label function-scenario vector, represented in multi-hot form. Finally, the joint training objective of MPP is defined as $\mathcal{L}_{\text{MPP}}$:

$$\mathcal{L}_{\text{MPP}} = \mathcal{L}_{\text{trig}} + \lambda \mathcal{L}_{\text{scenario}}, \quad (12)$$

where λ is the weight of the function-scenario prediction loss.

To better train PAR, we further fine-tune Qwen3.5-9B using SFT and GRPO. We construct a total of 8,876 multimodal and text data samples for SFT on Qwen3.5-9B, and we present the detailed construction process in Appendix A.3. Thus, given the input x_i, the model generates the structured output y_i. The two types of data are uniformly trained using the standard autoregressive cross-entropy loss, and the SFT loss is defined as:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{Z} \sum_{i,j} m_{i,j} \log P_\theta \left(y_{i,j} \mid x_i, y_{i,<j}\right), \quad (13)$$

where $y_{i,j}$ denotes the j-th token in the output of the i-th sample, and $y_{i,<j}$ denotes all tokens generated before the j-th token. Z denotes the total number of valid supervised tokens.

While SFT teaches PAR the structured response format, it may still be unstable on boundary cases

and imperfect in function selection or argument grounding. We therefore apply GRPO (Guo et al., 2025) after SFT to stabilize no-intervention decisions and improve executable assistance generation.

We construct GRPO prompts from rollouts of the SFT-trained PAR, denoted as the reference policy π_{ref}, and retain only rollout groups with non-trivial reward variance and intermediate correctness rates, leaving boundary no-intervention cases and intervention cases with disagreement in function selection or argument grounding. For each retained prompt x_i, GRPO samples N rollouts $\{\hat{y}_{i,j}\}_{j=1}^{N}$ and forms the group-relative advantage

$$\hat{A}_{i,j} = \frac{R(\hat{y}_{i,j}, x_i) - \mu_i}{\sigma_i + \delta}, \tag{14}$$

where μ_i and σ_i are the mean and standard deviation of the rollout rewards, and δ is a small numerical constant. We optimize the policy with the clipped GRPO objective and a KL penalty to π_{ref}. For a rollout $\hat{y}$, intervention outcomes are determined by whether the predicted and gold function sequences are empty. We use a structured reward

$$R(\hat{y}, x) = R_{\text{acc}}(\hat{y}, x) + R_{\text{fmt}}(\hat{y}) + R_{\text{cal}}(\hat{y}, x), \tag{15}$$

where R_{acc} scores the intervention decision, function-name sequence matching, and argument grounding, while R_{fmt} scores compliance with the required output schema. The calibration term R_{cal} adjusts the reward across intervention outcomes: it anchors correct silence on no-intervention cases, keeps false interventions penalized even when they are well formed, rewards complete gold-function coverage, and discourages redundant function calls. Full reward-component definitions and calibration details are provided in Appendix A.4.

4 Experiments

4.1 Experimental Settings

Training Details. We use the ProactiveMobile test set, which contains 3,660 samples (see Appendix A.2). The text and image encoders used in PRPF are BGE-small-zh-v1.5 (Luo et al., 2024) and CLIP ViT-B-32 (Radford et al., 2021). The base model used by PRPF is Qwen3.5-9B. Specific experimental details are provided in Appendix A.1.

Baselines. To verify the effectiveness of PRPF, we select three representative types of models for comparison experiments. **Closed-source models:** GPT-5.5 (OpenAI, 2026), o3 (OpenAI, 2025), Gemini-3.1-Pro (Google DeepMind, 2026), Claude-Opus-4.7 (Anthropic, 2026b), GLM-4.6V (Hong et al., 2025), Kimi-K2.5 (Team et al., 2026), and MiMo-2.5v (Xiaomi, 2026). **Open-source models:** TongUI-7B and Qwen3.5-9B (Zhang et al., 2025a; Bai et al., 2025). **Proactive intelligence models:** ProactiveMobile (7B), UI-TARS-7B-DPO+Proactive, and Qwen3.5-9B+Proactive are models SFT-trained using the same dataset and configuration.

Metrics. Following the work (Kong et al., 2026), we adopt three core metrics: function-name sequence accuracy (Type-Acc), success rate (SR), and false trigger rate (FTR). Type-Acc measures whether the predicted function-name sequence exactly matches the ground truth. SR evaluates the overall correctness of the prediction, while FTR quantifies the proportion of false proactive triggers among samples that do not require recommendation. The detailed calculation formulas are provided in Section B.3.

4.2 Overall Performance Analysis

Table 1 presents the overall performance comparison between PRPF and various baseline models on the ProactiveMobile test set.From these results, we draw conclusions:

1. PRPF achieves the best overall performance. Compared with the strongest fine-tuned baseline Qwen3.5-9B+Proactive, PRPF improves Type-Acc by 5.49 percentage points, improves SR by 6.59 percentage points, and reduces FTR by 6.28 percentage points. This suggests that, compared with standalone end-to-end VLMs, the two-stage PRPF better separates when to recommend from how to recommend, thereby improving recommendation success while suppressing false triggers.

2. FTR is significantly reduced. From the table, we observe that models without task-specific fine-tuning often exhibit high FTR. For example, GPT-5.5 reaches 23.84%. Fine-tuned models can better reduce false triggers, such as ProactiveMobile (7B), which reduces FTR to 13.76%. In contrast, our method directly reduces FTR to 7.21%, and even reduces it to 1.75% in the text setting. The extremely low false trigger rate verifies the effectiveness of the two-stage training and inference of PRPF, reducing the model's false trigger rate.

Model	Multimodal			Text			ALL		
	Type-Acc↑	SR↑	FTR↓	Type-Acc↑	SR↑	FTR↓	Type-Acc↑	SR↑	FTR↓
GPT-5.5	40.34	**18.40**	33.48	55.09	49.02	19.24	47.71	33.69	23.84
o3	39.63	17.25	41.45	51.31	44.53	26.89	45.46	30.87	31.61
Gemini-3.1-Pro	18.94	9.33	76.47	33.97	26.75	56.60	26.45	18.03	61.11
Claude-Opus-4.7	37.23	15.83	41.03	55.91	48.69	18.81	46.56	32.24	26.11
GLM-4.6V	36.25	11.74	74.45	29.98	20.19	67.04	33.12	15.96	69.33
Kimi-K2.5	25.33	9.33	30.02	39.50	35.56	32.51	32.40	22.43	31.64
MiMo-2.5v	32.10	11.63	39.45	39.08	34.54	37.06	35.58	23.07	37.87
TongUI-7B	1.75	0.33	**15.79**	8.15	8.04	18.58	4.95	4.18	18.10
UI-TARS-7B-DPO+Proactive	30.90	8.52	63.13	14.17	9.90	64.67	22.54	9.21	63.84
Qwen3.5-9B	8.30	2.95	71.52	8.15	5.69	77.13	8.22	4.32	75.30
ProactiveMobile (7B)	33.68	15.61	23.91	56.84	26.04	8.51	45.25	20.82	13.76
Qwen3.5-9B+Proactive	39.52	14.85	18.54	59.52	54.32	10.63	49.51	34.56	13.49
PRPF (Ours)	**40.83**	17.19	17.99	**69.20**	**65.15**	**1.75**	**55.00**	**41.15**	**7.21**

Table 1: Overall performance comparison on the ProactiveMobile test set. We report Type-Acc ↑, SR ↑, and FTR ↓ under multimodal, text, and overall settings. The best and second-best results are highlighted in bold and underlined, respectively. All scores are reported in percentage (%).

4.3 Ablation Study on Core Components

To further verify the effectiveness of each core component in PRPF, we conduct ablation studies on the same test set under the same settings according to the following four modules: **Overall Structure:** w/o MPP denotes removing MPP, where all samples directly enter PAR; w/o PAR denotes using Qwen3.5-9B for direct inference in the How stage. **MPP Structure:** w/o Slow Channel denotes removing the slow channel from MPP, and w/o Fast Channel denotes removing the fast channel from MPP. **MPP Functionality:** w/o Compression denotes that MPP only provides the recommendation judgment, while the full function set is input into PAR; w/o Recommend denotes that MPP does not perform recommendation judgment and only provides the function compression results. **PAR Training Strategy:** w/o SFT denotes removing SFT from PAR, and w/o GRPO denotes removing GRPO from PAR.

Model	Multimodal		Text		ALL	
	SR ↑	FTR ↓	SR ↑	FTR ↓	SR ↑	FTR ↓
Full PRPF (Ours)	**17.19**	17.99	**65.15**	**1.75**	**41.15**	**7.21**
- w/o MPP	15.56	22.01	61.16	4.22	38.33	10.38
- w/o PAR	6.06	60.64	60.89	3.33	33.44	18.46
- w/o Slow Channel	16.59	20.88	62.91	3.44	39.73	9.38
- w/o Fast Channel	16.54	**17.73**	64.33	2.02	40.41	7.32
- w/o Compression	16.65	19.11	64.50	2.02	40.55	7.78
- w/o Recommend	15.88	21.28	62.75	3.87	39.29	9.74
- w/o GRPO	16.05	48.63	63.35	2.81	39.67	15.56
- w/o SFT	4.91	61.60	60.18	4.37	32.51	18.61

Table 2: Ablation study of PRPF.

As shown in Table 2, MPP and PAR provide complementary benefits, as removing either module decreases ALL SR and increases FTR. Within MPP, the slow channel contributes more than the fast channel, since removing it leads to a larger drop in ALL SR and a greater increase in FTR. For MPP functionality, trigger gating plays a more important role than candidate function compression, as w/o Recommend causes more severe performance degradation than w/o Compression. For PAR training, SFT establishes the basic generation capability, whereas GRPO further improves the decision policy. Removing SFT substantially degrades both ALL SR and FTR, while removing GRPO mainly leads to a notable increase in ALL FTR.

4.4 Efficiency Analysis

We evaluate inference efficiency on the ProactiveMobile test set using three metrics: per-sample inference compute (TFLOPs), peak GPU memory (GB), and end-to-end latency (ms). The benchmark follows the ablation settings above, and detailed aggregation rules are provided in Appendix B.6. Figure 3 reports the resulting comparison.

Compared with the single-stage ProactiveMobile (7B) baseline, our PRPF (9B) reduces compute by 69.3% and end-to-end latency by 60.1%, at the cost of a 12.0% increase in peak memory because PRPF runs a stronger 9B PAR backbone on the gate-accepted fraction. Both MPP functions contribute to this saving, but in different ways. The intervention gate removes PAR calls for no-intervention observations, giving w/o Compression a 24.6% compute reduction and 29.8% latency re-

duction; peak memory rises by 26.7% because accepted contexts still use the full function pool on the 9B engine. By contrast, intent-candidate filtering reduces the candidate function pool from 63 functions to no more than 20 on average, so w/o Recommend lowers the cost of every PAR call, reducing compute by 57.1% and latency by 50.5% with the same 12.0% memory overhead. Thus, in the efficiency benchmark, candidate function-pool reduction has the larger standalone effect, while PRPF (9B) combines it with intervention gating so that heavy VLM-based assistance reasoning is invoked less often and with a shorter function-pool prompt when invoked.

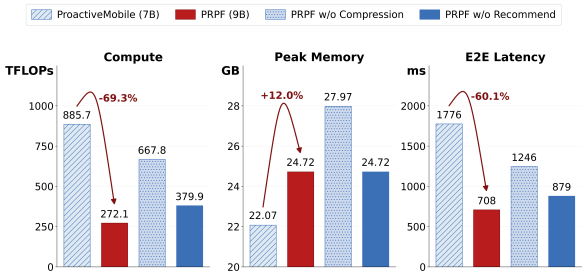


Figure 3: Inference efficiency comparison on the ProactiveMobile. Lower values indicate better efficiency.

4.5 Analysis of MPP

We select three groups of comparison experiments to verify the lightweight and plug-and-play characteristics of the perceptor. The three groups of experiments are ProactiveMobile (7B) vs. ProactiveMobile (7B) + MPP, GLM-4.6V vs. GLM-4.6V + MPP, and Qwen3.5-9B (SFT) Perceptor + PAR vs. PRPF. Among them, Qwen3.5-9B (SFT) Perceptor denotes that Qwen3.5-9B is trained with the same SFT method as ProactiveMobile (7B), serving as the perceptor to implement filtering and function selection, as shown in Table 3.

Model	Multimodal		Text		ALL	
	SR ↑	FTR ↓	SR ↑	FTR ↓	SR ↑	FTR ↓
ProactiveMobile (7B)	15.61	23.91	26.04	8.51	20.82	13.76
ProactiveMobile (7B) + MPP	12.61	21.95	62.20	**1.60**	37.38	8.08
GLM-4.6V	11.74	74.45	20.19	67.04	15.96	69.33
GLM-4.6V + MPP	13.92	68.75	62.91	4.15	38.39	19.97
Qwen3.5-9B (SFT) + PAR	17.08	19.55	62.58	3.55	39.80	8.98
Full PRPF (Ours)	**17.19**	**17.99**	**65.15**	1.75	**41.15**	**7.21**

Table 3: Plug-and-play effectiveness of MPP on PRPF.

The first two groups of comparisons demonstrate the plug-and-play property of MPP, showing that MPP can take effect once integrated into different models. The third group verifies that MPP achieves better results than the fine-tuned Qwen3.5-9B not only in terms of performance but also in terms of parameter size (0.1B vs. 9B).

4.6 Case Study

To localise PRPF's failures, we partition every test sample into one of five mutually exclusive outcomes under SR scoring and report the per-modality breakdown in Figure 4. PRPF achieves SR $= 1$ on 65.2% of TEXT samples but only 17.2% of Multimodal samples; the multimodal gap is mainly caused by *refusal* (34.8%), where the system stays silent on samples that require recommendation, and *function error* (24.1%), where PAR selects an incorrect intent. Detailed qualitative cases and a finer error decomposition (separating gate-side from PAR-side refusal, plus six sub-patterns of non-empty mismatch) are deferred to Appendix C.

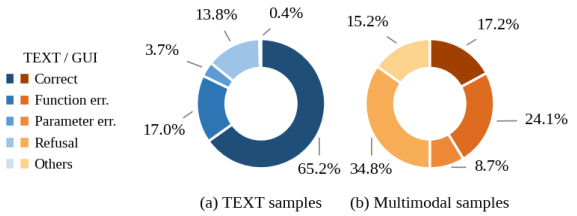


Figure 4: Per-modality breakdown of PRPF outcomes on the ProactiveMobile under SR.

5 Conclusion

In this paper, we introduce PRPF, a Pre-Reasoning Perception Framework for mobile proactive intelligence. PRPF explicitly decouples the two subtasks of when to recommend and how to recommend at the architectural level. Extensive experiments demonstrate the effectiveness of PRPF on the ProactiveMobile benchmark and show that it significantly reduces computational overhead. In addition, the gains brought by MPP across different reasoners further verify its strong plug-and-play capability. Our work provides an effective solution for building lightweight mobile proactive agents with low false-trigger rates and high success rates.

Limitations

MPP is trained on the 14 high-level intent scenarios defined in the ProactiveMobile benchmark. While this design suffices for the current function pool, extending PRPF to new domains or a significantly larger API space would require re-training (or at least fine-tuning) the perceiver to learn new intent-scenario distributions and update the Top-K

filtering vocabulary. Future work could explore continual-learning or prompt-based adapters to mitigate this re-training cost.

Multimodal understanding remains a bottleneck. Despite the overall improvement, absolute success rates on multimodal tasks are still modest (17.19% SR), and even the strongest baselines struggle to exceed 18% SR on ProactiveMobile's GUI screenshot setting. This gap suggests that current vision-language models still lack the fine-grained interface understanding required for ProactiveMobile assistance. Closing this gap likely demands stronger visual grounding, higher-resolution GUI encoders, or larger-scale multimodal pre-training beyond the scope of this work.

Ethics Statement

ProactiveMobile agents inherently require access to sensitive on-device signals, including GUI screenshots, user profiles, and interaction histories, which raises significant privacy concerns. We emphasize that all training and evaluation data used in this work are derived from the publicly available ProactiveMobile benchmark; no private user data or real-device traces were collected or used. For real-world deployment, PRPF must enforce explicit user consent and strict adherence to the principle of least privilege to prevent unauthorized surveillance or data leakage. Beyond initial consent, users should retain **granular control** over which on-device signals the agent may access, with the ability to audit and modify these permissions at any time. An **opt-out** mechanism must allow users to disable proactive interventions globally or per-application without degrading passive query functionality. Furthermore, **intervention personalization** should enable users to customize the frequency, timing, and sensitivity of proactive triggers—for example, suppressing suggestions during specific activities or setting daily intervention limits—thereby preventing behavioral over-inference and ensuring the agent respects individual boundaries.

References

Anthropic. 2026a. Claude sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6. Accessed: 2026-05-25.

Anthropic. 2026b. Introducing Claude Opus 4.7. Technical report, Anthropic.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Yuxiang Chai, Shunye Tang, Han Xiao, Rui Liu, and Hongsheng Li. 2026. Pira-bench: A transition from reactive gui agents to gui-based proactive intent recommendation agents. *arXiv preprint arXiv:2603.08013*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Liujianfeng Liujianfeng, Ang Li, Jian Luan, Bin Wang, Rui Yan, and 1 others. 2024. Mobilebench: An evaluation benchmark for llm-based mobile agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8813–8831.

Tom Fawcett. 2006. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874. ROC Analysis in Pattern Recognition.

Yuxuan Fu, Xiaoyu Tan, Teqi Hao, Chen Zhan, and Xihe Qiu. 2026. Prism: Festina lente proactivity–risk-sensitive, uncertainty-aware deliberation for proactive agents. *arXiv preprint arXiv:2602.01532*.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024. Cost-efficient large language model serving for multi-turn conversations with cachedattention. In *USENIX Annual Technical Conference*.

Google DeepMind. 2026. Gemini 3.1 pro model card. Technical report, Google DeepMind.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, and 1 others. 2025. Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Dezhi Kong, Zhengzhao Feng, Qiliang Liang, Hao Wang, Haofei Sun, Changpeng Yang, Yang Li, Peng Zhou, Shuai Nie, Hongzhen Wang, Linfeng Zhou, Hao Jia, Jiaming Xu, Runyu Shi, and Ying Huang. 2026. Proactivemobile: A comprehensive benchmark for boosting proactive intelligence on mobile devices. *Preprint*, arXiv:2602.21858.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*.

Fengrui Liu and 1 others. 2026a. Help without being asked: A deployed proactive agent system for on-call support with continuous self-improvement. In *Companion Proceedings of the 34th ACM Symposium on the Foundations of Software Engineering*.

Yichao Liu, Huawen Shen, Liu Yu, Shiyu Liu, Zeyu Chen, and Yu Zhou. 2026b. Drs-gui: Dynamic region search for training-free gui grounding. *arXiv preprint arXiv:2605.15542*.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, and 1 others. 2025. Proactive agent: Shifting llm agents from reactive responses to active assistance. In *International Conference on Learning Representations*, volume 2025, pages 47431–47457.

Kun Luo, Zheng Liu, Shitao Xiao, and Kang Liu. 2024. Bge landmark embedding: A chunking-free embedding method for retrieval augmented long-context large language models. *arXiv preprint arXiv:2402.11573*.

Sarthak Mehrotra, Sairam VC Rebbapragada, Mani Hemanth Reddy Bonthu, and Vineeth N Balasubramanian. 2025. ishift: Lightweight slow-fast gui agent with adaptive perception. *arXiv preprint arXiv:2512.22009*.

Deepak Nathani, Cheng Zhang, Chang Huan, Jiaming Shan, Yinfei Yang, Alkesh Patel, Zhe Gan, William Yang Wang, Michael Saxon, and Xin Eric Wang. 2026. Proactive agent research environment: Simulating active users to evaluate proactive assistants. *Preprint*, arXiv:2604.00842.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

OpenAI. 2025. Openai o3 and o4-mini system card. Technical report, OpenAI.

OpenAI. 2026. Gpt-5.5 system card. Technical report, OpenAI.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pages 406–441.

Fei Tang, Yongliang Shen, Hang Zhang, Siqi Chen, Guiyang Hou, Wenqi Zhang, Wenqiao Zhang, Kaitao Song, Weiming Lu, and Yueting Zhuang. 2025. Think twice, click once: Enhancing gui grounding via fast and slow systems. *arXiv preprint arXiv:2503.06470*.

Yuanbo Tang, Huaze Tang, Tingyu Cao, Lam Nguyen, Anping Zhang, Xinwen Cao, Chunkang Liu, Wenbo Ding, and Yang Li. 2026. Proagentbench: Evaluating llm agents for proactive assistance with real-world data. *Preprint*, arXiv:2602.04482.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, and 1 others. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*.

Zongru Wu, Pengzhou Cheng, Zheng Wu, Tianjie Ju, Zhuosheng Zhang, and Gongshen Liu. 2025. Smoothing grounding and reasoning for mllm-powered gui agents with query-oriented pivot tasks. *arXiv preprint arXiv:2503.00401*.

Xiaomi. 2026. MiMo-V2.5-Pro. Technical report, Xiaomi.

Zhifei Xie, Zongzheng Hu, Fangda Ye, Xin Zhang, Haobo Chai, Zihang Liu, Pengcheng Wu, Guibin Zhang, Yue Liao, Xiaobin Hu, and 1 others. 2026. Pask: Toward intent-aware proactive

agents with long-term memory. *arXiv preprint arXiv:2604.08000*.

Zhuoyan Xu, Khoi Duc Nguyen, Preeti Mukherjee, Saurabh Bagchi, Somali Chaterji, Yingyu Liang, and Yin Li. 2025. Learning to inference adaptively for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3552–3563.

Bufang Yang, Lilin Xu, Liekang Zeng, Yunqi Guo, Siyang Jiang, Wenrui Lu, Kaiwei Liu, Yixuan Li, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2026a. Proagent: Harnessing on-demand sensory contexts for proactive llm agent systems in the wild. *Preprint*, arXiv:2512.06721.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2026b. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *Advances in Neural Information Processing Systems*, 38:167509–167543.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. 2025a. Tongui: Building generalized gui agents by learning from multimodal web tutorials. *arXiv e-prints*, pages arXiv–2504.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025b. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

A Experimental Settings

A.1 Implement Details

The training hyperparameters of MPP, PAR supervised fine-tuning (SFT) and GRPO optimization are summarized in Table 4. All experiments are evaluated under the same setting, with the prompts, output format, and the list of available functions in the API pool kept consistent. PRPF is trained on NVIDIA H20 (96 GB HBM3) GPUs, consuming approximately 2,817 GPU-hours in total.

Parameter	Value
MPP	
Optimizer	AdamW
Learning Rate	1×10^{-4}
Trigger-Gating Threshold	0.78
Batch Size	8
Maximum Epochs	50
Early-Stopping Patience	10
Hidden Dimension	256
Scenario Loss Weight	$\lambda = 0.5$
PAR-SFT	
Optimizer	AdamW
Learning Rate	1×10^{-5}
LR Schedule	Cosine annealing
Warmup Ratio	0.1
Training Epochs	5
Batch Size	16
Gradient Accumulation Steps	2
PAR-GRPO	
Optimizer	AdamW
Learning Rate	5×10^{-7}
KL Coefficient	0.05
Rollouts per Prompt	$N = 8$
Batch Size	32
Training Epochs	1

Table 4: Training hyperparameters for MPP, PAR supervised fine-tuning, and GRPO optimization.

A.2 Datasets

ProactiveMobile (Kong et al., 2026) is a comprehensive benchmark dataset for mobile proactive intelligence. It is designed to evaluate whether a model can proactively infer users' latent needs from contextual information on mobile devices before users explicitly issue requests, and generate executable function-call sequences.

The dataset is constructed around four types of mobile contexts, including user profiles, device states, world information, and behavioral trajectories. Specifically, user profiles describe users' long-term habits, preferences, and basic attributes; device states reflect the current device and environmental status; world information provides external background such as weather, time, and holidays; and behavioral trajectories characterize the continuous interaction process between users and mobile devices. Meanwhile, the dataset also constructs a function pool containing 63 composite API functions. In terms of data format, ProactiveMobile contains both text scenarios and multimodal scenarios, where text scenarios use textual descriptions of behavioral trajectories, while multimodal scenarios use consecutive GUI screenshot sequences. Detailed dataset statistics are reported in Table 5. In addition, considering the diversity of user needs in real-world mobile scenarios, ProactiveMobile adopts a multi-answer annotation mechanism, where the same context may have multiple reasonable intent–function sequence annotations. Following the same protocol, this paper selects the function sequence with the highest F1 score.

Split	Data Type	Items	Intents	Images	Functions
Train	Multimodal	4438	8977	32418	9964
	Text	4438	4438	-	8259
Test	Multimodal	1832	3711	14341	4173
	Text	1828	2676	-	2266

Table 5: Statistics of the ProactiveMobile dataset, broken down by Train and Test splits and data modality. The table summarizes the composition of our benchmark, including the number of items, intents, UI images, and total functions.

A.3 SFT Data Construction

We construct two types of SFT data according to the sample modality: multimodal SFT data and text SFT data.

Multimodal SFT data construction. We decompose the recommendation generation process into four types of supervision signals: interface-state understanding, function-selection constraints, reasoning-process construction, and final recommendation generation. For recommendation samples, we use Claude Sonnet 4.6 for generation.(Anthropic, 2026a) Specifically, based on the GUI screenshot sequence, we first generate a UI Summary, which contains the current page stage, key interaction regions, and state changes. Then, combined with the user profile, device state, world information, and candidate functions, we generate the Thinking process containing Function Selection. The Function Selection includes candidate function-pool analysis, function-matching judgment, final function selection, and argument-source explanation. In this way, the model learns not only "what to recommend", but also "based on what context to recommend", "why to recommend at this moment", and "how to select executable functions from the candidate function pool". (Yang et al., 2026b; Kong et al., 2026) For non-recommendation samples, we do not construct the Thinking process and directly output the result.

Text SFT data construction. We decompose the recommendation generation process into three types of supervision signals: function-selection constraints, reasoning-process construction, and final recommendation generation. Except for the generation of UI Summary, all generation steps are consistent with those of multimodal SFT data.

A.4 GRPO Optimization Details

We use GRPO (Guo et al., 2025) as a post-SFT optimization stage for PAR. This appendix summarizes the rollout filtering, objective, and reward used for the GRPO stage.

Data construction. GRPO prompts are built from SFT-format prompts and rollouts of the SFT-trained PAR, which is used as the reference policy π_{ref}. Let $\mathcal{D}_{\text{roll}}$ denote the rollout set, where each case contains $N = 8$ sampled outputs and rule-based reward scores. We retain only rollout groups with non-trivial disagreement:

$$\text{std}(\{R_{i,j}\}_{j=1}^{N}) > 0.1, \qquad 0.1 \le \text{CR}_i \le 0.9. \tag{16}$$

Here CR_i is the rollout-level correctness rate under a case-specific intervention criterion. For intervention cases, correctness requires exceeding the TP-quality threshold used by the reward implementation, thereby filtering out mere triggering without sufficient function-name or argument correctness. For no-intervention cases, correctness means predicting an empty function-name sequence. Retained cases are weighted to emphasize function-correct but argument-weak cases and false-positive-prone no-intervention cases, then resized to 5,000 examples with at least 30% no-intervention records. During resizing, intervention records in the high-variance disagreement region, $\text{std}_i > 0.3$ and $0.1 < \text{CR}_i < 0.7$, are preferentially preserved, followed by a 95:5 stratified train/test split.

Reward. Let $S(y)$ denote the function-name sequence extracted from output y. For the supervised target y_i, $S_i^\star = S(y_i)$ is the gold function-name sequence in the SFT response; for rollout $\hat{y}_{i,j}$, $\hat{S}_{i,j} = S(\hat{y}_{i,j})$ is the predicted sequence. When multiple gold recommendations are available, $S^\star$ denotes the best-matched gold sequence selected by exact sequence accuracy and then function-name F1. In the reward formulas below, we omit instance and rollout indices and write $\hat{S}$ and $S^\star$ for readability.

The four intervention outcomes are defined by whether $\hat{S}$ and $S^\star$ are empty:

$$\mathbf{1}_{\text{TP}} = \mathbf{1}[|\hat{S}| > 0 \wedge |S^\star| > 0], \mathbf{1}_{\text{TN}} = \mathbf{1}[|\hat{S}| = 0 \wedge |S^\star| = 0], \tag{17}$$

$$\mathbf{1}_{\text{FP}} = \mathbf{1}[|\hat{S}| > 0 \wedge |S^\star| = 0], \mathbf{1}_{\text{FN}} = \mathbf{1}[|\hat{S}| = 0 \wedge |S^\star| > 0]. \tag{18}$$

The rollout reward is decomposed as

$$R(\hat{y}, x) = R_{\text{acc}}(\hat{y}, x) + R_{\text{fmt}}(\hat{y}) + R_{\text{cal}}(\hat{y}, x), \tag{19}$$

$$R_{\text{acc}} = w_T R_{\text{trig}} + (w_F R_{\text{func}} + w_A R_{\text{args}})\mathbf{1}_{\text{TP}}, \tag{20}$$

$$R_{\text{fmt}} = w_M R_{\text{schema}}, \tag{21}$$

$$R_{\text{cal}} = b_{\text{TN}}\mathbf{1}_{\text{TN}} + (R_{\text{fp}} + p_{\text{FP}})\mathbf{1}_{\text{FP}} + (b_{\text{cov}} + p_{\text{over}})\mathbf{1}_{\text{TP}}. \quad (22)$$

R_{trig} assigns quadrant-level intervention credit to TP/TN/FP/FN outcomes. R_{func} combines function-name set-F1 and longest-common-subsequence order consistency. R_{args} combines must-fill argument completeness and typed value similarity. R_{schema} scores output schema compliance, including tag order, function-field presence, JSON validity, and function-call structure. The calibration terms reward correct silence, keep false interventions net-penalized, reward full gold-function coverage, and penalize redundant function calls. Unparseable outputs receive only schema credit.

We set $w_T = 2.0$, $w_F = 3.0$, $w_A = 2.5$, and $w_M = 0.5$, with $b_{\text{TN}} = 6.0$, $p_{\text{FP}} = -1.5$, $b_{\text{cov}} = 0.2$, and $\lambda_o = 0.6$ for the over-prediction penalty. These constants are calibrated on development rollouts to preserve the intended ordering from malformed outputs, to valid no-intervention outputs, to partially correct TP outputs, and finally to fully correct executable recommendations. Our empirical ablations focus on the MPP/PAR architecture rather than treating reward constants as a separate contribution.

B Additional Experimental Results

B.1 MPP Hyperparameter Analysis

The main hyperparameter affecting the effectiveness of MPP is the threshold τ of the trigger-gating head, which directly determines how many samples need to be filtered. Since the goal of MPP is to filter out a large number of non-recommendation samples while retaining as many recommendation samples as possible, we mainly select two evaluation metrics: Recall and Specificity. Recall measures the proportion of recommendation samples that are correctly retained, i.e., avoiding missed recommendations, while Specificity measures the proportion of non-recommendation samples that are correctly filtered, i.e., effectively suppressing false triggers:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}, \text{Specificity} = \frac{\text{TN}}{\text{TN+FP}}. \quad (23)$$

We split the original training set into train/dev subsets with an 8:2 ratio. Threshold selection is performed exclusively on the dev split, while the test set is used only once for final evaluation. Following the standard idea of ROC operating-point selection(Fawcett, 2006), we formulate threshold selection as a constrained optimization problem: under the constraint of $\text{Recall} \geq 0.90$, i.e., the missed recommendation rate does not exceed 10%, we maximize Specificity. As shown in Figure 5, we perform grid search on the validation set over thresholds $\tau \in [0.05, 0.95]$ with a step size of 0.01. As τ increases, Recall monotonically decreases while Specificity monotonically increases, forming a trade-off curve between the two metrics. We select $\tau = 0.78$ as the deployment threshold, achieving $\text{Recall} = 90.42\%$ and $\text{Specificity} = 76.92\%$ on the dev split. This means that the gate can filter out approximately 77% of non-recommendation samples while retaining more than 90% of truly recommendation-needed samples. Among the thresholds satisfying the recall constraint, $\tau = 0.78$ provides the strongest filtering capability on the dev split.

B.2 Detailed Results and Experimental Analysis of MPP

For multimodal data, the processing procedure of MPP has been described in the main text. For text data, we only need to replace the GUI screenshot sequence with Trace Text, while keeping all other steps consistent.

The experiments in this paper are based on the results of MPP under the random seed of 42. The confusion matrix is shown in Table 6. MPP can filter out 1,320 samples, among which 1,081 samples are correctly filtered, achieving a correct filtering rate of 81.89%, demonstrating good performance. Moreover, a portion of the 169 cases will also be correctly handled in the subsequent PAR stage.

	Pred. Pos.	**Pred. Neg.**	**Total**
Actual Pos.	2171	239	2410
Actual Neg.	169	1081	1250
Total	2340	1320	3660

Table 6: Trigger-decision confusion matrix of MPP on the ProactiveMobile test set. Pos. and Neg. denote recommendation and non-recommendation, respectively.

Since MPP training can be affected by random initialization and other stochastic factors, we evaluate its stability across five random seeds: 7, 42, 123, 1234, and 3407. Each trained model is evaluated on the test set. Table 7 reports the seed-42 result used in the main experiments, as well as the five-seed

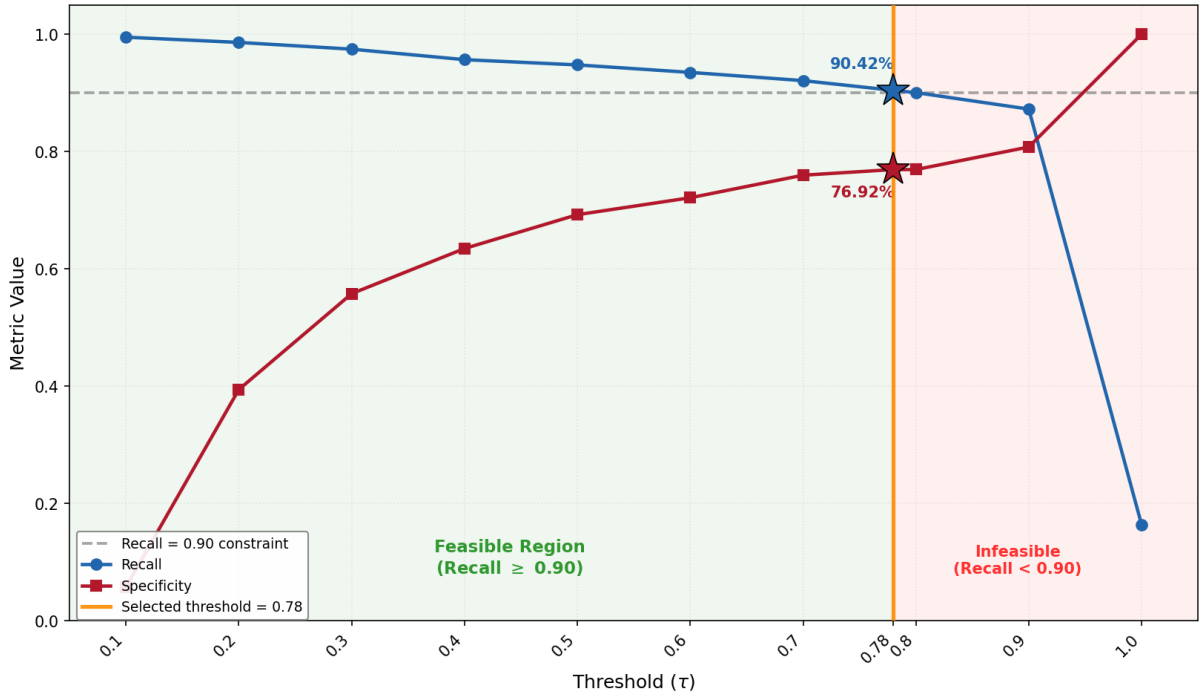


Figure 5: Sensitivity analysis of trigger-gate performance under different thresholds τ.

mean $\pm$ standard deviation. The results indicate that the trigger decision is stable and reliable: the standard deviation of Recall over five seeds is only ± 0.11, demonstrating that the model has very stable recognition ability for recommendation-needed samples under different random seeds. The filtering capability is also effective: Specificity reaches 83.81–86.48%, which means that MPP can filter out more than 83% of non-recommendation samples, significantly reducing the computational burden of the downstream decision model. The relatively large fluctuation of Specificity (± 2.45) is because the optimal threshold found by each seed differs greatly.Accuracy denotes the proportion of correctly predicted trigger decisions among all samples. F1 denotes the harmonic mean of Precision and Recall. scenario Top-5 Acc denotes whether the scenario ranked in the top 5 predicted by the model hits the ground-truth intent label among the correctly triggered samples (TP).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{24}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{25}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{26}$$

$$\text{scenario Top-5 Acc} = \frac{\sum_{c=1}^{C} N_c^{\text{correct}}}{\sum_{c=1}^{C} N_c}, \tag{27}$$

where C denotes the total number of scenario categories, N_c^{correct} denotes the number of samples in the c-th scenario category that are correctly hit by the Top-5 prediction, and N_c denotes the total number of samples in the c-th scenario category.

B.3 Expanded Results with Granular Metrics

Metric Calculation. Given a test set $\mathcal{D}$, let $S(y)$ denote the function-name sequence extracted from an output y. For a model prediction $\hat{y}$, we denote the predicted function-name sequence as $\hat{S} = S(\hat{y})$.

When multiple valid ground-truth recommendations are available, $\mathcal{G} = \{S_1^\star, \ldots, S_K^\star\}$ denotes the set of all valid gold function-name sequences. Each sequence consists of ordered function names.

For metrics that require matching against a single reference, $S^\star$ denotes the best-matched gold function-name sequence selected from $\mathcal{G}$ by maximizing exact sequence accuracy and then function-name F1.

For SR evaluation, the associated function arguments are additionally considered by the LLM judge.

Following ProactiveMobile (Kong et al., 2026), the calculation formulas for Type-Acc, SR, and FTR are shown in Eq. (28), Eq. (29), and Eq. (30), respectively.

Seed	Threshold	Accuracy	Recall	Specificity	Precision	F1	Intent Top-5 Acc
seed 42	0.78	88.85	90.08	86.48	92.78	91.41	90.96
5-seed mean ± std	0.59 ± 0.24	87.99 ± 0.78	90.16 ± 0.11	83.81 ± 2.45	91.49 ± 1.17	90.82 ± 0.53	87.95 ± 3.48

Table 7: Performance analysis of MPP under different random seeds.

$$\text{Type-Acc} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}\Big[\exists S^\star \in \mathcal{G}, \hat{S} = S^\star\Big], \tag{28}$$

where $|\mathcal{D}|$ denotes the number of test instances, $\mathcal{G}$ denotes the set of valid gold function-name sequences, and $\mathbb{I}[\cdot]$ is the indicator function that returns 1 if the condition is satisfied and 0 otherwise.

$$\text{SR} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}\Big[\exists S^\star \in \mathcal{G}, \hat{y} \equiv_{\text{func}} y^\star\Big], \tag{29}$$

where $y^\star$ denotes the ground-truth response corresponding to $S^\star$, and $\equiv_{\text{func}}$ denotes functional equivalence judged by the evaluation protocol.

Following ProactiveMobile (Kong et al., 2026), we employ Gemini-2.5-Pro as the LLM judge to evaluate functional equivalence.(Comanici et al., 2025) Specifically, when the predicted output $\hat{y}$ and a ground-truth response $y^\star$ are functionally equivalent, i.e., the selected functions and their corresponding arguments are semantically consistent, the SR score of this instance is assigned as 1; otherwise, the SR score is 0.

$$\text{FTR} = \frac{\sum_{i \in \mathcal{D}} \mathbb{I}\left[\mathcal{G} = \emptyset \wedge \hat{S} \neq \emptyset\right]}{\sum_{i \in \mathcal{D}} \mathbb{I}\left[\mathcal{G} = \emptyset\right]}, \tag{30}$$

where $\mathcal{G} = \emptyset$ denotes that the current instance does not require proactive recommendation, while $\hat{S} \neq \emptyset$ indicates that the model incorrectly triggers a recommendation.

Precision (P) and Recall (R). To provide a more forgiving set-based evaluation, we treat the prediction and ground truth as unordered sets of function names. Let $\hat{P}_{\text{set}} = \text{set}(\hat{S})$ and $G_{\text{set}} = \text{set}(S^\star)$.

Precision (P) measures the proportion of predicted functions that are relevant:

$$P = \frac{|\hat{P}_{\text{set}} \cap G_{\text{set}}|}{|\hat{P}_{\text{set}}|}. \tag{31}$$

Recall (R) measures the proportion of ground-truth functions that are successfully predicted:

$$R = \frac{|\hat{P}_{\text{set}} \cap G_{\text{set}}|}{|G_{\text{set}}|}. \tag{32}$$

B.4 Sensitivity Analysis of the Top-K Candidate-Pool Size in MPP

Beyond the gating threshold τ, a second design choice is the candidate-pool size K, which truncates MPP's ranked function list before it is passed to PAR. A natural intuition is that a larger K raises the chance the ground-truth function lies in the pool and should therefore monotonically improve the pipeline Strict Accuracy (SR). We test this by fixing the threshold at the value selected in §B.1 and sweeping $K \in \{1, \ldots, 10\}$ on the ProactiveMobile test set, reporting Top-K Recall and SR in Table 8. Top-K Recall counts an instance as recalled if at least one of its ground-truth function sequences has all of its scenes covered by the shortlist.

Recall saturates at $K = 5$ while SR is non-monotonic. Top-K Recall rises from 74.92% to 99.37% but flattens sharply after $K = 5$: at $K = 5$ recall is already 95.52%, and the marginal gain over the entire K=5→10 range is only 3.85 pp. SR, by contrast, is not monotonic — it peaks at 41.15% at $K = 5$ and drifts *down* to 40.11–40.98% across $K \in \{6, \ldots, 10\}$, contradicting the naive prior. We attribute this to distractor injection: once the ground-truth function is reliably in the shortlist, every additional slot is by construction almost always filled with an irrelevant function that occupies prompt budget and competes for PAR's attention during candidate scoring, rather than a missing-but-useful candidate. $K = 5$ is therefore both the saturation point of recall and the onset of distractor interference, and we adopt it as the default candidate-pool size in all main-paper experiments.

B.5 Detailed Results Analysis of Three Difficulty Levels

Following the difficulty division method of the ProactiveMobile dataset, we divide the dataset into three difficulty levels: L1, L2, and L3.

K	1	2	3	4	**5**	6	7	8	9	10
Top-K Recall (%)	74.92	85.19	90.66	93.69	**95.52**	97.02	97.84	98.47	98.93	99.37
SR (%)	39.54	39.89	40.90	40.19	**41.15**	40.87	40.49	40.98	40.55	40.11

Table 8: Sensitivity of MPP's top-K candidate-pool size on ProactiveMobile. The bolded column ($K = 5$) marks the saturation point of Top-K Recall, at which SR also peaks; we adopt it as the default.

Difficulty	Model	Multimodal						Text						ALL					
		Type-Acc↑	SR↑	FTR↓	F1↑	P↑	R↑	Type-Acc↑	SR↑	FTR↓	F1↑	P↑	R↑	Type-Acc↑	SR↑	FTR↓	F1↑	P↑	R↑
L1	GPT-5.5	52.54	22.03	57.14	61.72	63.14	62.15	83.78	81.08	7.76	87.65	88.22	87.90	74.01	62.60	13.36	79.53	80.37	79.84
	o3	**53.39**	22.88	74.07	60.03	61.44	59.75	80.70	76.45	14.16	82.05	82.63	81.79	72.15	59.68	20.73	75.16	76.00	74.89
	Gemini-3.1-Pro	36.44	14.41	93.33	47.85	48.02	49.58	58.69	55.60	38.21	63.35	63.45	64.03	51.72	42.71	41.85	58.50	58.62	59.51
	Claude-Opus-4.7	48.31	**23.73**	76.00	63.05	63.14	65.11	86.87	82.24	8.15	88.10	88.61	87.97	74.80	63.93	15.04	**80.26**	**80.64**	**80.81**
	GLM-4.6V	50.85	**23.73**	91.30	**63.14**	**64.41**	63.84	43.63	37.84	58.57	45.61	46.08	45.62	45.89	33.42	61.80	51.10	51.81	51.33
	Kimi-K2.5	32.20	13.56	40.63	49.69	47.02	59.61	67.95	64.87	24.54	70.21	69.88	71.24	56.76	48.81	26.61	63.78	62.73	67.60
	MiMo-2.5v	50.85	13.56	48.28	62.54	62.01	**65.40**	59.46	57.53	32.55	63.26	62.77	65.19	56.76	43.77	34.44	63.03	62.53	65.25
	TongUI-7B	1.70	0.00	**0.00**	1.70	1.70	1.70	8.11	8.11	32.26	8.11	8.11	8.11	6.10	5.57	30.30	6.10	6.10	6.10
	UI-TARS-7B-DPO+Proactive	36.44	6.78	75.00	43.08	44.92	42.23	21.24	18.92	53.85	22.27	22.78	22.01	26.00	15.12	57.66	28.78	29.71	28.34
	Qwen3.5-9B	14.41	5.09	54.55	17.94	18.08	18.64	5.41	4.63	86.49	6.98	7.14	6.89	8.22	4.78	82.35	10.41	10.57	10.57
	ProactiveMobile (7B)	32.20	10.17	42.31	38.84	40.25	38.56	82.24	37.07	6.82	83.45	83.46	83.59	66.58	28.65	10.57	69.49	69.94	69.50
	Qwen3.5-9B+Proactive	38.98	5.09	21.62	42.09	43.22	41.53	86.87	84.17	3.65	87.65	87.65	87.65	71.88	59.42	6.25	73.39	73.74	73.21
	PRPF (Ours)	43.22	13.56	32.26	50.00	50.85	50.00	**92.28**	**90.35**	**0.90**	**93.31**	**93.44**	**93.44**	**76.92**	**66.31**	**4.74**	79.75	80.11	79.84
L2	GPT-5.5	42.41	**19.58**	38.24	55.91	**55.85**	58.62	64.14	58.51	20.33	71.89	72.81	72.17	54.08	40.48	24.27	64.49	64.96	65.89
	o3	41.60	19.09	54.76	53.93	55.49	54.30	58.09	51.06	31.26	64.28	65.68	63.98	50.45	36.25	36.26	59.48	60.96	59.49
	Gemini-3.1-Pro	21.04	10.28	81.25	39.64	37.65	45.73	34.60	28.27	62.28	47.10	47.29	49.35	28.32	19.94	64.88	43.64	42.83	47.68
	Claude-Opus-4.7	37.36	15.99	54.47	**56.26**	55.61	**61.26**	64.14	58.37	19.20	71.20	72.15	71.53	51.74	38.75	26.47	64.29	64.50	66.77
	GLM-4.6V	40.95	12.07	83.67	53.61	55.74	53.59	29.26	21.10	70.75	37.89	39.80	37.94	34.67	16.92	73.29	45.17	47.18	45.19
	Kimi-K2.5	22.35	7.34	36.03	41.67	37.81	52.72	47.26	44.30	32.20	55.78	54.91	59.12	35.73	27.19	33.10	49.25	46.99	56.16
	MiMo-2.5v	31.81	11.75	46.40	47.09	46.47	51.58	45.29	42.05	38.85	54.75	54.02	57.56	39.05	28.02	40.48	51.20	50.53	54.79
	TongUI-7B	1.96	0.49	20.00	1.96	1.96	1.96	11.81	11.81	13.40	11.81	11.81	11.81	7.25	6.57	14.29	7.25	7.25	7.25
	UI-TARS-7B-DPO+Proactive	35.89	9.46	64.91	43.47	45.35	42.74	14.21	11.11	64.24	16.95	18.00	16.46	24.25	10.35	64.52	29.23	30.67	28.63
	Qwen3.5-9B	8.81	2.94	70.73	13.82	13.72	14.76	9.28	7.17	75.42	11.90	12.34	11.98	9.06	5.21	74.55	12.79	12.98	13.27
	ProactiveMobile (7B)	34.75	17.62	23.78	43.39	44.13	44.02	69.06	32.49	9.07	72.34	72.60	72.68	53.17	25.60	12.61	58.93	59.42	59.41
	Qwen3.5-9B+Proactive	36.54	13.87	25.15	40.62	41.60	40.24	70.04	64.98	11.11	73.64	74.97	73.08	54.53	41.31	14.67	58.35	59.52	57.87
	PRPF (Ours)	**43.23**	17.46	**19.64**	50.24	51.33	50.27	**81.72**	**78.90**	**1.10**	**86.69**	**87.20**	**87.04**	**63.90**	**50.45**	**5.48**	**69.81**	**70.59**	**70.02**
L3	GPT-5.5	37.88	17.35	29.14	51.06	**52.98**	51.54	38.93	31.47	26.45	53.10	55.58	53.08	38.34	23.53	27.86	51.95	54.12	52.22
	o3	37.06	15.62	33.00	49.74	52.20	49.64	36.83	29.49	29.70	52.03	55.42	51.19	36.96	21.70	31.46	50.74	53.61	50.32
	Gemini-3.1-Pro	15.90	8.27	72.96	33.55	32.88	37.78	25.99	16.78	64.80	47.32	48.46	50.53	20.32	12.00	68.45	39.58	39.70	43.37
	Claude-Opus-4.7	35.97	14.90	33.13	**51.82**	51.35	**55.27**	39.74	30.54	27.10	57.32	59.55	58.50	37.62	21.75	30.41	**54.23**	**54.94**	**56.68**
	GLM-4.6V	32.06	10.26	69.14	45.47	48.62	44.85	26.46	14.10	68.47	43.86	47.27	43.77	29.61	11.95	68.83	44.77	48.03	44.38
	Kimi-K2.5	26.25	9.99	26.35	41.00	38.44	48.93	24.48	19.46	40.43	44.21	41.42	52.20	25.47	14.14	32.36	42.40	39.74	50.36
	MiMo-2.5v	30.25	11.35	35.87	42.24	42.83	43.90	27.77	21.35	37.66	45.86	45.34	50.14	29.16	15.73	36.64	43.83	43.93	46.63
	TongUI-7B	1.64	0.27	**14.29**	1.64	1.64	1.64	5.13	4.90	20.00	5.14	5.13	5.25	3.17	2.30	18.42	3.17	3.17	3.22
	UI-TARS-7B-DPO+Proactive	27.52	8.17	61.45	35.92	39.03	34.60	12.01	6.18	78.21	16.26	17.41	15.81	20.73	7.30	65.29	27.31	29.56	26.37
	Qwen3.5-9B	7.36	2.73	73.45	12.41	13.05	12.66	8.04	4.78	72.73	12.84	13.41	13.21	7.66	3.62	73.13	12.60	13.21	12.90
	ProactiveMobile (7B)	33.24	15.08	22.63	39.79	40.77	40.21	39.04	17.37	8.75	44.50	45.28	44.85	35.78	16.08	16.08	41.85	42.74	42.24
	Qwen3.5-9B+Proactive	**41.24**	16.44	15.26	45.34	47.18	44.48	42.54	36.48	14.74	50.91	53.67	49.65	41.81	25.22	15.03	47.78	50.03	46.75
	PRPF (Ours)	39.24	**17.44**	16.10	44.87	46.13	44.69	**51.87**	**46.15**	**3.20**	**60.47**	**62.26**	**60.51**	**44.77**	**30.02**	**9.68**	51.70	53.19	51.62
Avg	GPT-5.5	40.34	**18.40**	33.48	53.37	**54.59**	54.59	55.09	49.02	19.24	65.30	66.90	65.44	47.71	33.69	23.84	59.33	60.74	60.01
	o3	39.63	17.25	41.45	51.80	53.89	51.85	51.31	44.53	26.89	61.05	63.27	60.50	45.46	30.87	31.61	56.42	58.58	56.17
	Gemini-3.1-Pro	18.94	9.33	76.47	36.51	35.45	41.20	33.97	26.75	56.60	49.51	50.13	51.99	26.45	18.03	61.11	43.00	42.78	46.59
	Claude-Opus-4.7	37.23	15.83	41.03	**54.03**	53.53	**57.91**	55.91	48.69	18.81	67.08	68.57	67.74	46.56	32.24	26.11	60.55	61.04	**62.82**
	GLM-4.6V	36.25	11.74	74.45	49.33	52.02	49.00	29.98	20.19	67.04	41.79	44.20	41.77	33.12	15.96	69.33	45.56	48.11	45.39
	Kimi-K2.5	25.33	9.33	30.02	41.78	38.78	50.88	39.50	35.56	32.51	52.39	50.70	57.59	32.40	22.43	31.64	47.08	44.73	54.23
	MiMo-2.5v	32.10	11.63	39.45	45.17	45.28	47.85	39.08	34.54	37.06	51.79	51.19	55.16	35.58	23.07	37.87	48.48	48.23	51.50
	TongUI-7B	1.75	0.33	**15.79**	1.75	1.75	1.75	8.15	8.04	18.58	8.16	8.15	8.21	4.95	4.18	18.10	4.95	4.95	4.97
	UI-TARS-7B-DPO+Proactive	30.90	8.52	63.13	38.91	41.52	37.81	14.17	9.90	64.67	17.38	18.40	16.94	22.54	9.21	63.84	28.15	29.97	27.39
	Qwen3.5-9B	8.30	2.95	71.52	13.24	13.60	13.75	8.15	5.69	77.13	11.65	12.11	11.83	8.22	4.32	75.30	12.44	12.85	12.79
	ProactiveMobile (7B)	33.68	15.61	23.91	40.93	41.86	41.38	56.84	26.04	8.51	60.84	61.32	61.16	45.25	20.82	13.76	50.88	51.58	51.26
	Qwen3.5-9B+Proactive	39.52	14.85	18.54	43.55	45.06	42.87	59.52	54.32	10.63	64.96	66.77	64.15	49.51	34.56	13.49	54.24	55.90	53.50
	PRPF (Ours)	**40.83**	17.19	17.99	47.00	48.17	46.90	**69.20**	**65.15**	**1.75**	**75.32**	**76.38**	**75.49**	**55.00**	**41.15**	**7.21**	**61.14**	**62.26**	61.18

Table 9: Detailed performance comparison using granular, set-based metrics on the ProactiveMobile test set. We report function-name sequence accuracy (Type-Acc↑), success rate (SR↑), false trigger rate (FTR↓), F1-Score↑, Precision (P↑), and Recall (R↑) across different difficulties and modalities. Best results are in bold, and second-best results are underlined. All scores are reported in percentage (%).

- **Level 1 (Easy):** Correctly solved by 4–5 out of 5 reference models.
- **Level 2 (Medium):** Correctly solved by 2–3 out of 5 reference models.
- **Level 3 (Hard):** Correctly solved by 0–1 out of 5 reference models.

The results of PRPF and all baseline models are shown in Table 9, which also provides more

detailed insights:

1. PRPF maintains stable advantages across different difficulty levels. As the task difficulty increases from L1 to L3, PRPF achieves ALL SR of 66.31%, 50.45%, and 30.02%, respectively, all outperforming all baselines. In particular, PRPF still maintains a clear lead on L2 and L3, indicating that the gains of the model do not come from overfitting to easy samples, but are also effective in medium- and high-difficulty proactive recommendation scenarios.

2. PRPF can generate higher-quality function sets. The ALL F1 and Precision of PRPF are the highest among all baselines, while Recall is the second highest. This indicates that the improvement of PRPF does not only come from the increase of a small number of exactly matched samples, but also from the overall improvement in the selection quality of function sets, enabling the model to more accurately cover the key functions required by users' latent needs.

3. PRPF demonstrates strong cross-modal applicability. Across different difficulty levels, PRPF achieves higher ALL SR than the strongest baseline, Qwen3.5-9B+Proactive, in both text and multimodal settings. In the text setting, PRPF improves ALL SR from 54.32% to 65.15% compared with Qwen3.5-9B+Proactive, while reducing FTR from 10.63% to 1.75%. In the multimodal setting, PRPF also improves SR from 14.85% to 17.19%. These results indicate that PRPF provides stable proactive perception and content generation capabilities under both text and multimodal conditions.

4. Multimodal tasks remain a bottleneck. Although PRPF achieves an SR of 17.19% on multimodal tasks, outperforming Qwen3.5-9B+Proactive, the absolute value remains low. Even GPT-5.5, which performs best on multimodal tasks, achieves an SR of only 18.40%. This further demonstrates the challenge of the ProactiveMobile task and indicates that stronger interface understanding capability is still required in the future to achieve a breakthrough in this domain.

B.6 Efficiency Benchmark Details

This appendix details the evaluation protocol for the efficiency analysis in Section 4.4. All reported values are computed on the ProactiveMobile test split and aggregated per test sample after applying the routing policy of each configuration.

Compute Estimation. For each PAR invocation, we estimate FLOPs from the actual prompt and generation lengths observed during inference. Let T_p denote the prompt length, T_g the generated length, N_p the number of model parameters, and L and d the number of transformer layers and hidden size. The PAR FLOPs are estimated as

$$F_{\text{PAR}} = 2N_p(T_p + T_g) + 4LdT_p(T_p + T_g). \quad (33)$$

The first term approximates parameter-dominated dense computation, while the second term accounts for the attention cost induced by long multimodal prompts.

MPP is measured as an end-to-end front-end, including text encoding, image encoding, and the slow–fast multimodal fusion network. Its measured per-sample cost is 23.58 GFLOPs, 14.16 ms latency, and 1.51 GB peak memory. Let r denote the empirical fraction of test samples routed from MPP to PAR:

$$r = \frac{N_{\text{trig}=1}}{N} = \frac{2{,}340}{3{,}660} = 0.639. \quad (34)$$

Aggregation Across the Test Set. Compute and latency are reported as per-sample averages after applying each system's routing policy. The single-stage baseline invokes the proactive VLM on every sample with the full function pool. The w/o Compression variant keeps the MPP intervention gate but passes the full function pool to PAR, so PAR is invoked only on the r fraction accepted by the gate. The w/o Recommend variant disables the intervention shortcut but keeps MPP's scenario-candidate filtering, so PAR is invoked on every sample with the candidate function pool. Full PRPF combines both mechanisms: the intervention gate skips no-intervention observations, and scenario-candidate filtering reduces the function pool for accepted contexts.

Method	Compute	Latency
ProactiveMobile (7B)	$F_{\text{PAR}}^{\text{full}}$	$T_{\text{PAR}}^{\text{full}}$
w/o Compression	$F_{\text{MPP}}+rF_{\text{PAR}}^{\text{full}}$	$T_{\text{MPP}}+rT_{\text{PAR}}^{\text{full}}$
w/o Recommend	$F_{\text{MPP}}+F_{\text{PAR}}^{\text{comp}}$	$T_{\text{MPP}}+T_{\text{PAR}}^{\text{comp}}$

Table 10: Aggregation rules for per-sample compute and latency in the efficiency benchmark.

Memory and Latency Measurement. Peak memory is reported as the maximum active stage because MPP and PAR are invoked sequentially: for the single-stage baseline ProactiveMobile (7B),

$M = M_{\text{PAR}}$; for the PRPF variants, $M = \max(M_{\text{MPP}}, M_{\text{PAR}})$. This convention reflects the deployed cascade in which the lightweight pre-reasoning stage and the heavy PAR stage are not treated as one simultaneous forward pass. A conservative co-resident upper bound can be obtained by summing the two stage memories, but the main efficiency table reports the sequential peak.

Latency is measured with the same batched inference path used by the evaluation pipeline. For each configuration, PAR latency is the total wall-clock inference time divided by the number of samples that enter PAR; for observations rejected by the intervention gate, the latency aggregation includes the measured MPP cost and no PAR forward pass.

C Failure Case Analysis

This appendix provides the failure-case analysis referenced in Section 4.6 and complements the multimodal limitation discussed in Limitations. All numbers reported here are computed on the ProactiveMobile test set ($N = 3{,}660$) and correspond to the PRPF row of Table 1: ALL Type-Acc $= 55.00$, SR $= 41.15$, FTR $= 7.21$. The analysis is conducted on the same predictions, so the aggregate numbers in this section can be reconciled with the main table by composition.

C.1 Error Categorisation

We use the trigger gate output $g \in \{0, 1\}$ from MPP, the predicted function-name sequence $\hat{S}$ from PAR, and the best-matched gold sequence $S^\star$ defined in Appendix A.4. A prediction is counted as a Type-Acc error when $\hat{S}$ is not order- and length-equal to $S^\star$. Errors are partitioned into four mutually exclusive categories:

- **Non-empty mismatch**: $|\hat{S}| > 0$ and $\hat{S} \neq S^\star$.
- **Post-gate abstention**: $g = 1$ and $|\hat{S}| = 0$ while $|S^\star| > 0$. PAR was invoked but the policy declined to recommend.
- **Gate misfire**: $g = 0$ while the best-matched gold intent requires recommendation. PAR is never invoked, so no recovery is possible.
- **Parse failure**: $\hat{S}$ cannot be parsed back into a function-call sequence.

Table 11 reports the absolute counts. Roughly equal mass falls into *non-empty mismatch* (45.8%) and *post-gate abstention* (42.0%): once PAR is invoked, it is almost as likely to over-abstain as it is to call a wrong function. Gate misfire is the smallest bucket (12.2%) but is fully attributable to MPP and cannot be repaired downstream.

Category	Count	% of errors
Non-empty mismatch	751	45.8
Post-gate abstention	689	42.0
Gate misfire	200	12.2
Parse failure	7	—
Total Type-Acc errors	1,647	100.0

Table 11: Error categorisation for PRPF on the ProactiveMobile test set. Parse failures are reported as orphan items because they fall outside the four well-formed prediction quadrants.

The non-empty mismatch bucket can be further decomposed by comparing the predicted set against the best-matched gold set (Table 12). *Off-scene* mismatch dominates (40.6% of non-empty mismatches): PAR returns a syntactically valid call sequence whose function names share no overlap with $S^\star$, indicating that the failure is at the intent-selection level rather than the argument-filling level.

Sub-pattern	Count	%
Off-scene ($\hat{S} \cap S^\star = \emptyset$)	305	40.6
Spurious ($S^\star = \emptyset, \hat{S} \neq \emptyset$)	124	16.5
Missed steps ($\hat{S} \subsetneq S^\star$)	154	20.5
Extra steps ($S^\star \subsetneq \hat{S}$)	80	10.7
Reordering (same set, wrong order)	56	7.5
Partial overlap	32	4.3
Total	751	100.0

Table 12: Sub-patterns of non-empty mismatch errors. The dominant failure mode is off-scene misrouting, not argument-level error.

C.2 Where Errors Concentrate: Modality and Gating Path

To localise the bottleneck, we cross-tabulate Type-Acc by modality (TEXT vs Multimodal) and gate decision (g=0 silenced vs g=1 routed to PAR), shown in Table 13.

Two observations matter for the multimodal-bottleneck argument. First, the g=0 silenced path is concentrated on TEXT (93.6% of silenced samples are TEXT), where the trigger decision is comparatively easy and acc reaches 0.866. The high overall TEXT accuracy reported in the main table is largely attributable to this silence dividend rather

Modality	g=0 (silenced)		g=1 (routed to PAR)	
	n	acc	n	acc
TEXT	1,235	0.866	593	0.329
Multimodal	85	0.588	1,747	0.400

Table 13: Type-Acc cross-tabulated by modality and gating path. Most TEXT samples ($1{,}235/1{,}828 = 67.6\%$) are silenced by the gate at high accuracy. The bottleneck shifts to PAR once a sample is routed: g=1 accuracy is below 0.41 for both modalities.

than to stronger TEXT recommendation by PAR. Second, on the g=1 path TEXT accuracy (0.329) is in fact *lower* than Multimodal accuracy (0.400): the easy TEXT cases have already been gated out, and what remains for PAR is the harder TEXT residual. This indicates that further gains require improving PAR's reasoning on the routed subset, not pushing the gate to filter more aggressively.

C.3 Scene-Level Concentration

The 14 high-level intent scenes used by MPP differ markedly in error profile. We group them into three substantive clusters (Table 14); two scenes with very small n (Logistics & Delivery, $n = 54$; Smart Home, $n = 10$) are excluded from clustering to avoid noise.

The off-scene mismatch sub-pattern is highly concentrated in Cluster II: Travel & Lodging (60), Shopping (47), and Personal Management (44) together account for $151/305 = 49.5\%$ of all off-scene errors, suggesting that the residual errors in Cluster II are disproportionately rooted in confusable functions across closely related lifestyle scenes (e.g. booking, ordering, ride-hailing) rather than in interface understanding.

C.4 A Reference Multimodal Success Case

To complement the failure statistics above with a concrete upper-bound reference, Table 15 reproduces a single multimodal sample on which PRPF achieves SR= 1 in a Financial Services scene—the behaviour the system already exhibits when the gate routes correctly and PAR's `<thinking>` and `<function_selection>` both align with the gold function. The table reproduces the complete benchmark inputs—user profile U, device state D, world context W, and trace H—together with PAR's parsed structured output as `<ui_summary>` → `<thinking>` → `<recommendations>`. The `<thinking>` block reproduces both the free-form reasoning and the embedded `<function_selection>` sub-block. The case is drawn from `huggingface_strict_correctcase.json` and is rendered as a single-column `longtable` at the end of this appendix (Table 15), flowing across pages without a continuation header. The original ProactiveMobile data is in Chinese; we render all natural-language content in English here for readability, with function names, parameter keys, and structural tags preserved verbatim.

C.5 Implications

The category- and slice-level numbers above isolate two complementary residual problems for PRPF. (i) On the modality axis, g=1 Multimodal samples remain the dominant absolute error mass and the multimodal limitation reported in the Limitations section is quantitatively concentrated in Cluster I, where TEXT accuracy already exceeds 0.80. Closing this gap will mainly require stronger GUI grounding inside PAR. (ii) On the intent-selection axis, the off-scene sub-pattern in Cluster II indicates that PAR's failures in travel-/shopping-/management-style scenes are routing failures rather than argument-level failures. This points to expanding the function set seen during training and refining the candidate-pool compression in MPP, rather than further reward shaping at the GRPO stage.

D Prompts for LLM Agents

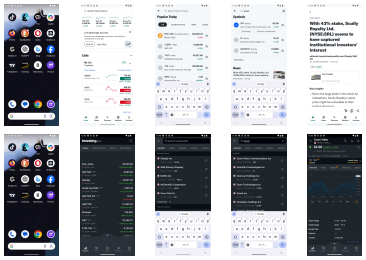
At inference time, the trained PAR model (Section 3.4) consumes a single user-message prompt template assembled by the eval-data builder: each ProactiveMobile sample is rewritten into one turn that bundles the per-sample context (U, D, W), the MPP-predicted intent scenes, and the MPP-restricted function pool, and the resulting message is wrapped with the standard Qwen-3 chat template before being fed to vLLM. Only samples for which the MPP trigger gate predicts $g = 1$ are routed through this prompt; gate-silenced samples are short-circuited to the empty-recommendation output without a forward pass through PAR. The sampler-side conversation is built with a fixed system message `"You are a helpful assistant."` followed by a single user message that concatenates a Chinese task header, an XML-tagged context bundle, the MPP-predicted scenes, the available function pool, and a structured-output instruction tail; for multimodal samples the leading `<image>` place-

Cluster	Representative scenes	n	acc	$\bar{\Delta}_{T-M}$
I. Multimodal-hard	Sports & Health Entertainment & Media Office Work Social Communication	1,200	0.645	$+0.41$
II. Domain-knowledge-hard	Travel & Lodging Food & Dining Transportation	997	0.431	$+0.18$
III. Balanced	Device & System Management Shopping Personal Management Content Creation Financial Services	1,399	0.557	$+0.24$

Table 14: Scene-level clusters with mean Type-Acc and mean TEXT$-$Multimodal gap $\bar{\Delta}_{T-M}$. Cluster II is the largest absolute drag on overall accuracy and shows a small modality gap, indicating that even the TEXT branch underperforms in these scenes. Cluster I has high overall accuracy but the largest modality gap, so further gains specifically require stronger GUI grounding.

holders are stripped from the textual content and the corresponding screenshot files are attached as `image_url` entries so that vLLM places the image tokens at the same position relative to the textual context as during SFT. A sample at the end of the appendix document the assembled prompt for the multimodal and text-only branches; the original prompts are in Chinese, and we render them in English here for readability while preserving JSON keys, structural tags, and function names verbatim.

Table 15: Case study (a): Multimodal — Match. Scene: Financial Services. Sample id `40590079`.

User Profile (U)	A 35-year-old technology-industry professional with seven years of investing experience; holds shares in Zoom, Apple, Tesla, Amazon, and other tech companies...
Device Status (D)	Time 12:48; WiFi connected; battery sufficient. Multiple finance apps installed: Yahoo Finance, Investing.com, Bloomberg, TradingView. Over the past three months the user has averaged 15+ daily opens of finance apps...
World Information (W)	Currently in market hours; NASDAQ $+2.03\%$, S&P 500 $+1.02\%$; market sentiment positive. Tech sector broadly stronger; multiple cloud-computing and e-commerce stocks up...
Behavioral Trajectories (H)	 Frames 1–10: Yahoo Finance → search "Zoom" → Simply Wall St. article → Investing.com → Zoom Video (NASDAQ: ZM) ticker page.
`<ui_summary>`	Page progression: from the home screen the user opens Yahoo Finance and checks "My List" (containing AMZN, AAPL, TSLA), enters "Zoom" in the search bar, taps a Simply Wall St...

`<thinking>`	The user is currently viewing the Zoom Video (NASDAQ: ZM) ticker page on Investing.com, with price \$62.00 ($+1.57\%$). Earlier he searched Zoom on Yahoo Finance and read a Simply Wall St. article on Scully Royalty Ltd., then switched platforms. This sequence indicates multi-platform information gathering at the pre-decision stage...
	`<function_selection>`
	Candidate function pool. `search_financial_info`, `create_financial_analysis_view`, `create_portfolio`, `manage_budget_and_plans`, `bank_service_assistance`, ...
	Step-by-step reasoning.
	• `search_financial_info`: supports `search_target="analysis_articles"`, `search_keywords="Zoom"`, `location="US"`; directly matches. **Match.**
	• `create_financial_analysis_view`: builds a chart view, not a search. **Excluded.**
	• ...
	Selected function and parameters. `search_financial_info` with `search_target="analysis_articles"`, `search_keywords="Zoom"`, `location="US"`.
	`</function_selection>`
	`</thinking>`

`<recommendations>`	*Instruction:* "Recommend deep analysis articles or research reports on Zoom."
	Function call (parsed):
	`[{`
	`"name":`
	`"search_financial_info",`
	`"parameters": {`
	`"search_target":`
	`"analysis_articles",`
	`"search_keywords": "Zoom",`
	`"filter_criteria": "",`
	`"sort_order": "",`
	`"category": "",`
	`"include_analysis": "",`
	`"location": "US"`
	`}`
	`}]`

Outcome	Type-Acc match; the LLM judge pass, **SR**$= 1$**.**

Prompt $\mathcal{P}_{\text{inf-GUI}}$: PAR Inference Prompt (Multimodal/GUI)

You are an intelligent assistant. Based on the user profile (profile), device status (phone), environmental information (world), and screenshots, analyze the user's current behavior and determine whether a recommendation is needed.
User profile
Device status
Environmental information
predicted_scenes {List of predicted scene probabilities, different for each sample}
available_functions {List of available functions and parameter definitions, different for each sample}
Please output in the following order:
ui_summary: Based on the screenshots, describe the current stage of the user interface, key interaction areas, state changes, and the current stage of the workflow. Include key identifiable entities in the screenshots, such as specific app names, page titles, contacts, file names, etc.
thinking: Conduct sufficient step-by-step reasoning covering the following aspects:
a) Behavior recognition: Describe what the user is doing based on the screenshots and extract key entities such as app names, file names, contacts, page content, etc.;
b) Intent analysis: Combine the user profile (profile), device status (phone), and environmental information (world) to analyze the user's deeper intent and the current task stage (just started, in progress, completed);
c) Need assessment: Determine whether the user currently has any unmet needs or whether there is an optimizable next step. If the task has been completed and there is no obvious follow-up need, clearly conclude that no recommendation is needed;
d) If a recommendation is needed, explain what to recommend, why it should be recommended at this moment rather than letting the user complete it themselves, and the key parameter information required for the recommendation, such as target app, operation object, recipient, etc.
recommendations: Based on the selected functions and parameters in function_selection, generate the final function call(s), which may include 0, 1, or multiple functions. The function names and parameter values should be consistent with the reasoning conclusion in function_selection, in the following format: ["instruction": "Recommendation instruction", "function": ["name": "FunctionName1", "parameters": ..., "name": "FunctionName2", "parameters": ...]]
If there is no recommendation, function_selection is not needed in thinking; directly output: No recommendation