

# Why Act Without Asking?

## Evaluating Mobile GUI Agents under Ambiguous Instructions

Yeongjae Cho, Jungwuk Park, Tae-Min Choi,  
Chiyoun Park, Mincheol Kim, Kookhoi Kim

Samsung Research

{yeongjae.cho, jungwuk.park, tae\_min.choi,  
chiyoun.park, mc154.kim, khoi.kim}@samsung.com

### Abstract

Mobile GUI agents often act confidently under ambiguous or incomplete user instructions, a common situation in real-world use. Even when explicitly offered a “call the user” action, open-weight agents ask for clarification in only 0.1% of cases on average, instead guessing the user’s intent and proceeding with actions. We name this failure *Act-without-Asking*. Existing step-level benchmarks largely miss this failure mode because they credit any ground-truth-matching action regardless of asking. We introduce MID-Bench, a step-level ambiguity benchmark constructed by rewriting clear instructions into ambiguous ones, with a strict clarification protocol: a user simulator answering only what is asked, and a scoring rule crediting a step only when the agent both asks an appropriate question and emits the correct final action. The *Hard* subset applies stricter masking to the action-critical detail in the instruction, making it harder to succeed without asking. Across four open-weight GUI agents, train-free cascades in which a gate decides whether to act or ask the user for clarification substantially improve ambiguous-task performance while preserving accuracy on clear instructions; LoRA fine-tuning on MID-Bench further combines effectively with these cascades.

## 1 Introduction

Recent progress in compact multimodal models has made it practical to deploy GUI agents directly on mobile devices, where an agent predicts UI actions from a user instruction and the current screen state (Hong et al., 2024; Qin et al., 2025; Xu et al., 2026; Wang et al., 2024; Zhang et al., 2025; Wu et al., 2025; Liu et al., 2024). Existing mobile GUI-agent benchmarks evaluate such agents either through step-level action matching (Li et al., 2024; Cheng et al., 2024; Lu et al., 2025; Rawles et al., 2023; Zhang et al., 2024a) or through task completion in interactive environments (Toyama et al.,

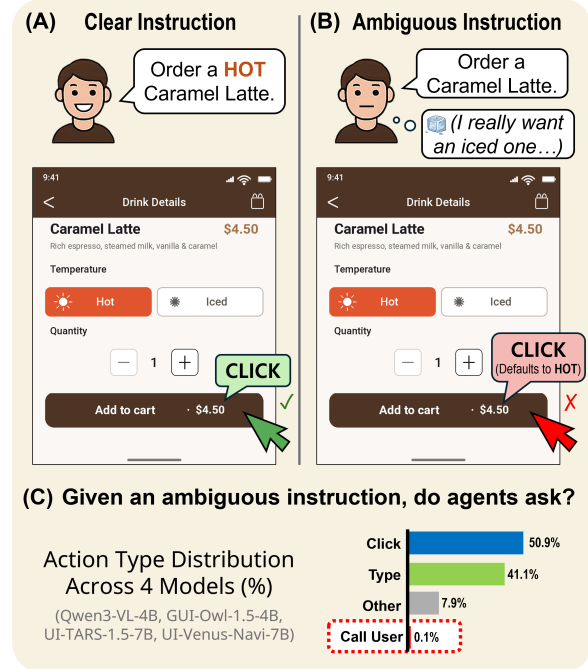


Figure 1: **Act-without-Asking in mobile GUI agents.** (a) Given a fully specified instruction, the agent emits the action matching the user’s request. (b) Given an ambiguous instruction, the agent commits to a plausible but incorrect GUI action instead of asking for clarification. (c) Under ambiguous instructions, four open-weight GUI agents almost never ask questions, with clarification accounting for only 0.1% of emitted actions on average.

2021; Rawles et al., 2025), typically under clear and fully specified user instructions.

In practice, however, user instructions are often ambiguous or incomplete (Rao and Daumé III, 2018; Aliannejadi et al., 2019; Min et al., 2020). In interactive GUI environments, acting on an incorrect interpretation of the user’s intent can lead to unintended outcomes, such as messaging the wrong recipient, modifying the wrong setting, or purchasing an unintended item (e.g., Fig. 1b). Reliable GUI agents should therefore recognize when an instruction is underspecified, stop acting, and ask the

user for clarification. Yet we find that current GUI agents rarely do so. Existing GUI-agent training data (Li et al., 2024; Lu et al., 2025) contains few clarification examples: most agents have rarely encountered such scenarios during training, and even agents that natively include clarification behavior (e.g., UI-Venus (Gu et al., 2025)) have little signal as to when to invoke it. Agents thus default to their training distribution, continuing with plausible GUI actions rather than asking under ambiguous instructions; clarification behavior accounts for only 0.1% of emitted actions on average across four open-weight agents (Fig. 1c). We refer to this failure mode as **Act-without-Asking**.

Prior work has begun to study ambiguity and clarification in GUI agents (Kong et al., 2025; Sun et al., 2026, 2025; Cheng et al., 2025; Hao et al., 2025), but measuring clarification at each step has been difficult because existing benchmarks do not support step-level evaluation of whether the agent should ask for clarification before acting. As a result, agents can receive credit for correct actions even when they proceed without explicitly resolving ambiguity. These studies therefore evaluate clarification mainly at the trajectory or dialogue level, making it difficult to isolate Act-without-Asking from appropriate clarification behavior.

To address this gap, we introduce **Mobile Instruction Disambiguation Benchmark (MID-Bench)**: 900 ambiguous-instruction steps (600 *Base*, 300 *Hard*) paired with a strict user simulator that answers only the agent’s specific question, enabling step-level evaluation of clarification. Success requires the agent to both ask an appropriate clarification question and emit the correct final action given the simulator’s response. The *Base* split rewrites instructions under four ambiguity types; the *Hard* subset further restricts these to type-action steps where the typed text is verifiably removed verbatim (e.g., “type ‘molenstraat’ ” → “type my home address”), making it harder to succeed without asking. We further design **train-free clarification cascades** in which a gate decides, at each step, whether to execute the frozen base agent’s action or ask the user for clarification. Across multiple open-weight GUI agents, these cascades substantially improve performance under ambiguous instructions while preserving accuracy on clear instructions. Finally, we show that lightweight LoRA fine-tuning on MID-Bench improves clarification behavior and composes effectively with train-free clarification gates.

## 2 Related Work

**GUI agent benchmarks and instruction ambiguity.** Existing mobile GUI-agent benchmarks evaluate either step-level action prediction (Li et al., 2024; Cheng et al., 2024; Lu et al., 2025) or end-to-end task completion in interactive environments (Toyama et al., 2021; Rawles et al., 2025), but largely assume that user instructions are fully specified. More recent work begins to incorporate ambiguity and clarification into GUI-agent evaluation: AmbiBench (Sun et al., 2026) categorizes instructions by ambiguity level, MobileWorld (Kong et al., 2025) introduces an *ask\_user* action in an interactive emulator, and Fairy (Sun et al., 2025) evaluates user-requirement completion under interactive assistance. However, these benchmarks evaluate clarification primarily at the trajectory or dialogue level. In contrast, our focus is Act-without-Asking: the step-level decision of whether the agent should stop and ask at the moment ambiguity becomes action-blocking.

### Mitigations for clarification under uncertainty.

Prior work on ambiguous instructions has explored various ways of inducing language models to ask for clarification, including clarification question elicitation (Rao and Daumé III, 2018; Aliannejadi et al., 2019), prompted or learned ambiguity detectors (Kuhn et al., 2022; Zhang et al., 2024b), disagreement across sampled completions (Cole et al., 2023; Zhang and Choi, 2025; Farquhar et al., 2024), and confidence estimates derived from the model’s forward pass (Hendrycks and Gimpel, 2016; Kadavath et al., 2022; Lin et al., 2022; Tian et al., 2023). Similar ideas have recently appeared in GUI agents, either by post-training the base policy to ask clarification questions (Cheng et al., 2025; Suri et al., 2025; Wang et al., 2025) or by augmenting frozen agents with additional reasoning modules (Hao et al., 2025; Sun et al., 2025). In this work, we focus on lightweight **train-free clarification cascades** that wrap frozen GUI agents with clarification gates, enabling direct comparison of different clarification signals under a common benchmark.

## 3 Motivating Observations

We begin by analyzing how current GUI agents behave under ambiguous instructions. We evaluate four open-weight GUI agents: Qwen3-VL-4B (Bai et al., 2025), GUI-Owl-1.5-4B (Xu et al.,

|                  | Empty          | Ambiguous              |                |
|------------------|----------------|------------------------|----------------|
|                  | Ask $\uparrow$ | Recognition $\uparrow$ | Ask $\uparrow$ |
| Qwen3-VL-4B      | 2.4            | 16.8                   | 0.1            |
| GUI-Owl-1.5-4B   | 0.7            | <b>43.8</b>            | 0.0            |
| UI-TARS-1.5-7B   | <b>12.3</b>    | 14.6                   | 0.2            |
| UI-Venus-Navi-7B | 0.0            | 4.2                    | 0.1            |

Table 1: **Recognition–action gap.** Empty vs. ambiguous instructions. *Ask*: percentage of tasks where the agent invokes `call_user` in action mode. *Recognition*: percentage of ambiguous tasks labeled AMBIGUOUS in classification mode.

2026), UI-TARS-1.5-7B (Qin et al., 2025), and UI-Venus-Navi-7B (Gu et al., 2025). Throughout the paper, agents are explicitly instructed to ask for clarification under ambiguity, and the action space includes a dedicated clarification action, denoted `call_user` (prompts in App. A.1, A.2).

### 3.1 Agents continue acting under maximal uncertainty

To probe the extreme case where asking is most clearly warranted, we replace only the user-instruction field with an empty string while keeping the full agent setup otherwise unchanged, including the system prompt, action space, and clarification mechanism. Even under this maximal uncertainty setting, the agents rarely invoke `call_user`: ask rates are only 2.4% (Qwen3-VL-4B), 0.7% (GUI-Owl-1.5-4B), 12.3% (UI-TARS-1.5-7B), and 0.0% (UI-Venus-Navi-7B) on empty-instruction steps (Tab. 1, left). Instead, the models continue emitting GUI actions even when clarification is explicitly available and encouraged by the prompt. This suggests a strong prior toward action continuation even under maximal uncertainty.

### 3.2 Recognizing ambiguity does not lead to clarification

To localize Act-without-Asking, we evaluate each model in two modes on the same ambiguous instruction and screen input. In *Recognition*, the model classifies the (screen, instruction) pair as CLEAR / AMBIGUOUS / NO\_TASK without producing an action (prompt in App. A.3). In *Action*, the model generates the next GUI action on the same ambiguous instruction and screenshot. This setup distinguishes two failure modes: low recognition implies failure to perceive ambiguity, whereas high recognition but low `call_user` invocation rate implies that ambiguity signals fail to propagate into

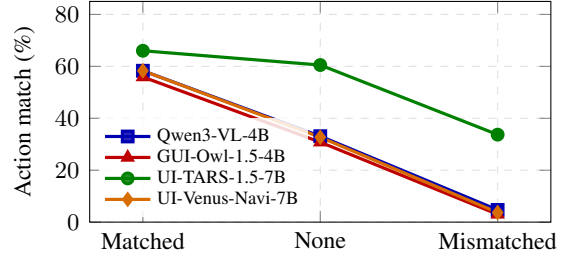


Figure 2: **Action match under instruction perturbations.** *Matched*: original instruction; *None*: instruction removed; *Mismatched*: instruction replaced with a goal from a different episode.

action selection.

Tab. 1 (right) shows strong evidence for the latter. GUI-Owl-1.5-4B, the strongest recognizer, labels 43.8% of ambiguous tasks as AMBIGUOUS yet invokes `call_user` on 0% of them, a 43.8 pp gap. Qwen3-VL-4B recognizes 16.8% and asks on 0.1%; UI-TARS-1.5-7B recognizes 14.6% and asks on 0.2%; UI-Venus-Navi-7B recognizes 4.2% and asks on 0.1%. Several models recognize ambiguity in classification mode, yet almost none ask for clarification during action generation.

### 3.3 Actions are weakly conditioned on instructions

If ambiguity recognition fails to influence action selection (§3.2), how strongly are actions conditioned on the instruction itself? We present the same screenshot under three instruction variants: *matched* (the original goal), *none* (empty instruction), and *mismatched* (a goal from a different episode), then measure whether the predicted action matches the ground truth (Fig. 2).

Even when the instruction is removed entirely (*none*), action match remains surprisingly high across all models, with Qwen3-VL-4B, GUI-Owl-1.5-4B, UI-TARS-1.5-7B, and UI-Venus-Navi-7B retaining 33.1%, 30.8%, 60.5%, and 32.7% match rates, respectively. Under *mismatched*, UI-TARS-1.5-7B still matches the original action on 33.7% of steps despite being conditioned on instructions from a different task, whereas Qwen3-VL-4B, GUI-Owl-1.5-4B, and UI-Venus-Navi-7B drop sharply to 4.7%, 3.0%, and 3.7%. This suggests a possible mechanism behind Act-without-Asking: when agents rely more strongly on plausible actions suggested by the current screen state than on instruction semantics, ambiguous or incomplete instructions may fail to trigger `call_user`, leading the agent to continue acting instead.

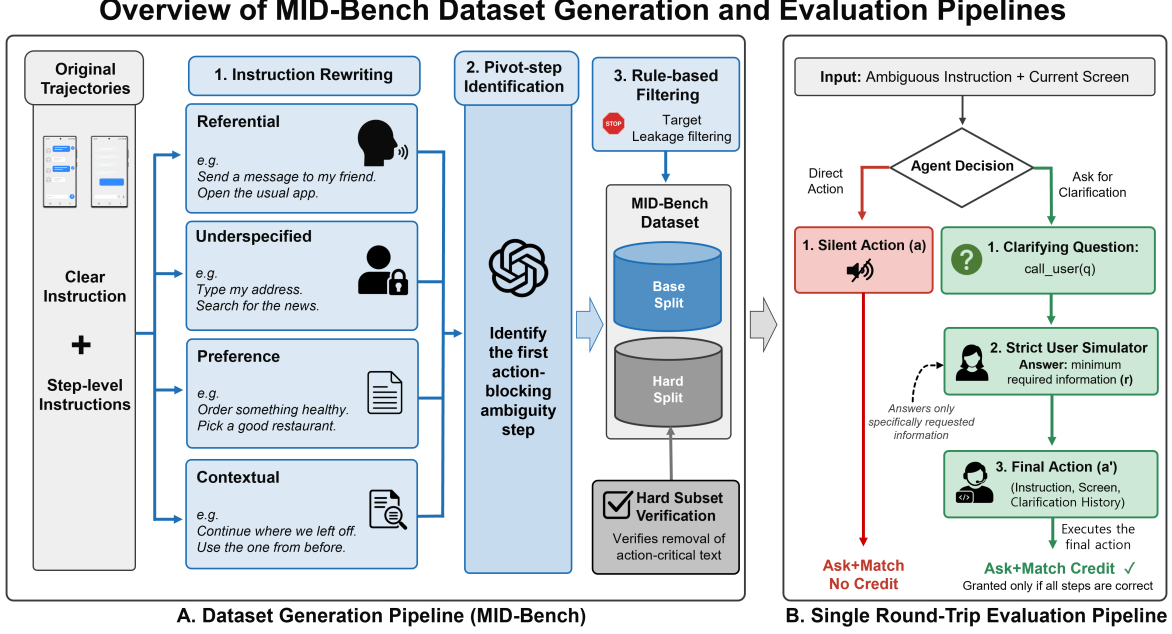


Figure 3: **MID-Bench pipeline.** (A) Construction of ambiguous instructions and the stricter *Hard* subset. (B) Single-round ask→answer→final-action evaluation under a strict simulator, where success requires both clarification and a correct final action (Ask+Match).

## 4 MID-Bench

The observations in §3 suggest that current GUI agents often act under ambiguity without clarification, while existing step-level benchmarks still reward such behavior. To explicitly evaluate this failure mode, we introduce MID-Bench, a benchmark for evaluating clarification behavior in mobile GUI agents under ambiguous instructions.

### 4.1 Dataset

**Splits.** To construct ambiguous-instruction scenarios for mobile GUI agents, we build on test trajectories from AndroidControl (AC) (Li et al., 2024) and GUI-Odyssey (Ody) (Lu et al., 2025). MID-Bench consists of two ambiguity splits requiring clarification before acting (collectively, the *Ambig* pool): **Base** ( $n=600$ ; 400 AC + 200 Ody) and **Hard** ( $n=300$ ; 1:1), a stricter masked-token subset. We additionally include a **Clear** split ( $n=300$ ) containing unambiguous instructions and ground-truth actions from the same AC+Ody trajectories, allowing us to measure standard action accuracy alongside ambiguity handling.

**Construction pipeline.** The *Base* split is built in three stages, with *Hard* added as a stricter variant (Fig. 3A; full prompts in App. C). **(1) Instruction rewriting.** Each instruction is rewritten under one of four ambiguity types: *Referential* (vague names),

*Underspecified* (missing parameters, e.g., “type my home address” for a specific string), *Preference-dependent* (subjective criteria), and *Contextual* (assumed prior context), balanced 1:1:1:1 and preserving the GT action. **(2) Pivot-step identification.** Given the original and rewritten instructions together with the trajectory’s step list, an LLM identifies the first step at which the masked detail becomes action-critical; trajectories with no such step are dropped. **(3) Rule-based filtering.** Three deterministic rules drop candidates that confound the protocol: (R1) GT action in {scroll, long\_press, wait, navigate\_back}, and (R2/R3) the rewritten instruction contains the GT type text (*typed-full-leak*) or open\_app target name (*app-name-leak*) as a substring. R1–R3 yield the released *Base* split. **Hard subset.** *Hard* is constructed from the same source datasets under stricter masking constraints: only type-action steps whose typed text appears verbatim in the original instruction enter the pool, and the rewrite must remove that text verbatim (e.g., “type ‘molenstraat’ ” → “type my home address”); we auto-verify that every typed-text token of length  $\geq 4$  is absent from the rewrite. As a result, the action-critical typed string is verifiably absent from the instruction, making the correct type action significantly harder to recover without first asking the user. We instantiate all construction-time LLM calls with GPT-4o (Achiam et al., 2023).

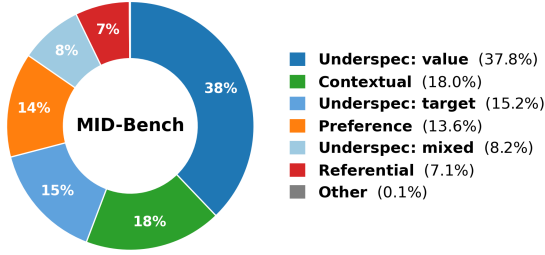


Figure 4: **Ambiguity-category composition of MID-Bench** (confirmed-ambiguous subset). *Underspec* splits into *value*, *target*, and *mixed*.

**Validation.** We conduct an internal validation study on the released Ambig pool to evaluate (Q1) whether the rewritten instruction is genuinely ambiguous given the screenshot and intended next action, and (Q2) a 7-way ambiguity category label. Annotators were shown the screenshot, rewritten instruction, and a translucent overlay of the GT next action. Q1 confirms ambiguity validity on 92.2% (830/900) of examples, indicating that the benchmark items consistently leave the next action underdetermined from the provided instruction and screen context alone. Over the confirmed-ambiguous subset, *Underspecified* cases appear more frequently than the intended 1:1:1:1 category balance (Fig. 4). See App. D for prompts and details.

## 4.2 Protocol, simulator, and metrics

**Single round-trip.** Given the instruction and screenshot (Fig. 3B; worked examples in Fig. 5), (1) the model emits action  $a$  or `call_user( $q$ )`; (2) if  $a$ , score  $a$  against GT and stop; (3) if `call_user( $q$ )`, a strict simulator returns answer  $r$ ; (4) the model then emits the final action on the same instruction with the `call_user( $q$ )/User:  $r$`  pair appended to history, producing  $a'$ , scored against the original GT. To approximate realistic deployment, we restrict `call_user` to a single round: subsequent `call_user` emissions score 0.

**Strict simulator.** Existing user simulators (Sun et al., 2026; Kong et al., 2025) condition on a hidden goal and refuse off-task or low-level UI queries, but answer any on-task clarification regardless of its specificity. We tighten this: our simulator answers only questions specific enough to map to a single piece of the original (unmasked) goal (e.g., “Which book?” → “The Queen’s Gambit.”), while an underspecified question receives a non-informative

response (full prompt in App. A.4). We further examine whether the simulator can be replaced at lower cost: a small 4B model preserves the reported scores (App. E).

**Metrics.** We report three metrics, each tied to a distinct capability. **Clear** measures baseline action competence on fully specified instructions: action accuracy on the Clear split, irrespective of asking behavior. Action correctness, here and throughout, follows the step-level matching rules of ScreenSuite (Roucher and Mahla, 2025). **Ask+Match** is our primary metric for Act-without-Asking on the Ambig splits (*Base/Hard*). A step is credited iff the agent both fires `call_user` and subsequently executes the correct action; matching actions taken without asking receive a score of 0. Ask+Match thus operationalizes ambiguity-aware acting by requiring agents to explicitly resolve ambiguity before acting. As a relaxed counterpart, **Match-Only** mirrors existing step-level benchmarks by crediting any action that reaches the ground truth regardless of whether the agent fired `call_user`. We include it to separate pure task completion from ambiguity-aware acting. Because it ignores the asking decision, Match-Only cannot measure Act-without-Asking: it conflates correctness achieved through clarification with correctness obtained without resolving ambiguity.

## 5 Train-Free Clarification Cascades

**Overview.** We wrap a frozen base agent with the ask→answer→final-action interaction protocol introduced in §4.2. At each step, the wrapper either keeps the baseline action or triggers clarification before producing the final action, with the choice controlled by a *gate*. The gate signal defines the sole architectural difference across variants: we compare three train-free choices below while holding clarification generation, the strict simulator, and the final execution stage fixed, so comparisons isolate the effect of the gate signal.

**PGC (Prompted-Gate Cascade).** A separate verbalized prompt asks the base model whether the current instruction is sufficiently clear to act on (Kuhn et al., 2022; Zhang et al., 2024b). The gate fires iff the response is negative. Because the decision is produced directly as a verbalized yes/no judgment, no threshold tuning is required (prompt in App. A.5).

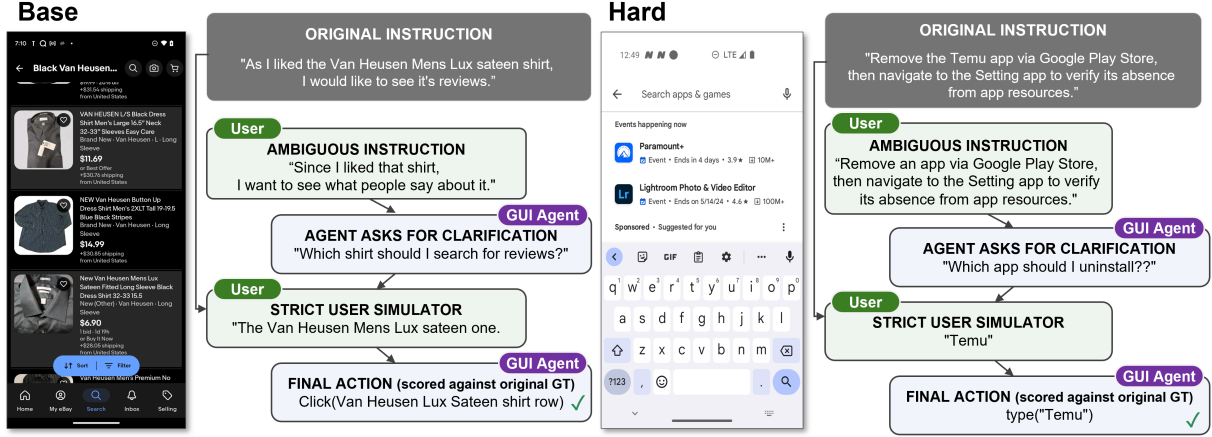


Figure 5: **Illustrative single-round ask → answer → final action interactions under MID-Bench.** The original instruction is rewritten into an ambiguity-inducing variant that leaves the next action underdetermined. A successful agent asks a clarification question, receives only the minimum information required from the strict user simulator, and then executes the original ground-truth action.

**SGC (Sampling-Gate Cascade).** We draw  $k=5$  stochastic decodes of the baseline action and group them into action-equivalence classes before computing semantic entropy (Farquhar et al., 2024). For example, click actions are grouped by spatial buckets, typed text is normalized before comparison, and low-level actions such as swipe are treated as equivalent. The gate fires iff the resulting entropy exceeds  $\tau_{\text{SGC}}$  (App. L).

**LGC (Logit-Gated Cascade).** We reuse the baseline forward pass and fire the gate iff the mean token log-probability of the baseline action falls below  $\tau_{\text{LGC}}$  (Hendrycks and Gimpel, 2016). Unlike PGC and SGC, LGC requires no additional forward pass beyond the baseline action generation.

## 6 Experiments

### 6.1 Setup

**Models and cascades.** We use four frozen, greedy-decoded bases, Qwen3-VL-4B (a general-purpose VLM), GUI-Owl-1.5-4B, UI-TARS-1.5-7B, and UI-Venus-Navi-7B, each wrapped with the three train-free cascades of §5 (PGC, SGC, LGC). The *baseline* row in Tab. 2 is the same frozen base under the same protocol; so each cascade row differs only by its gate signal. Thresholds for SGC and LGC are F1-optimized on a held-out validation split and transferred unchanged. On Qwen3-VL-4B we additionally report a **LoRA SFT** (Hu et al., 2022) row and its composition with the train-free gates (App. H).

**Splits and metrics.** We score on two ambiguity splits, *Base* (base Ambig pool,  $n=600$ ) and *Hard* ( $n=300$ ), under two scoring rules. **Match-Only** mirrors prior step-level benchmarks by crediting any non-call\_user action matching GT, regardless of whether the agent asks for clarification. **Ask+Match** is our primary metric for Act-without-Asking: it additionally requires the agent to fire call\_user before emitting the final action. The Match-Only – Ask+Match gap on a baseline row therefore measures the amount of credit assigned to actions taken without clarification. We additionally report **Clear** to check for performance degradation on unambiguous instructions; should-ask F1 is reported in App. J.

### 6.2 Main results

**Prior evaluation hides Act-without-Asking.** Under Match-Only, baselines land at .30–.42 on *Base* Ambig, below Clear (.42–.52) but well above zero, a level that prior step-level benchmarks would read as competent. The number does not reflect Act-without-Asking: empirical call\_user emission is at most 2/900 across *Base* and *Hard*, and 0/300 on Clear, so essentially none of this credit follows clarification; the agent acted without asking and happened to land on GT. The blind spot becomes obvious on *Hard*, where the typed string is verifiably masked and the same baselines collapse to .08–.11. Our cascades nevertheless improve over baselines on every split, reaching .39–.54 on *Base* (+.02–.17) and .13–.56 on *Hard* (+.03–.47): the lift is largest exactly where Match-Only exposed

| Model                                 | Method   | Clear Split | Base Ambig |           | Hard Ambig |           |
|---------------------------------------|----------|-------------|------------|-----------|------------|-----------|
|                                       |          | Clear       | Match-Only | Ask+Match | Match-Only | Ask+Match |
| Train-free clarification cascades     |          |             |            |           |            |           |
| Qwen3-VL-4B                           | baseline | .468        | .406       | .000      | .107       | .000      |
|                                       | +PGC     | .461        | .430       | .134      | .223       | .150      |
|                                       | +SGC     | .480        | .470       | .171      | .367       | .273      |
|                                       | +LGC     | .489        | .476       | .407      | .447       | .423      |
| GUI-Owl-1.5-4B                        | baseline | .518        | .415       | .000      | .100       | .000      |
|                                       | +PGC     | .529        | .447       | .150      | .267       | .203      |
|                                       | +SGC     | .522        | .541       | .408      | .513       | .460      |
|                                       | +LGC     | .508        | .537       | .478      | .467       | .397      |
| UI-TARS-1.5-7B                        | baseline | .419        | .297       | .000      | .077       | .000      |
|                                       | +PGC     | .402        | .418       | .384      | .437       | .430      |
|                                       | +SGC     | .393        | .433       | .397      | .443       | .433      |
|                                       | +LGC     | .403        | .432       | .357      | .440       | .440      |
| UI-Venus-Navi-7B                      | baseline | .466        | .370       | .000      | .097       | .000      |
|                                       | +PGC     | .539        | .510       | .435      | .513       | .507      |
|                                       | +SGC     | .466        | .389       | .051      | .127       | .037      |
|                                       | +LGC     | .542        | .536       | .536      | .563       | .563      |
| Supervised fine-tuning (SFT) contrast |          |             |            |           |            |           |
| Qwen3-VL-4B<br>(LoRA)                 | baseline | .493        | .361       | .333      | .323       | .230      |
|                                       | +PGC     | .479        | .416       | .391      | .333       | .247      |
|                                       | +SGC     | .472        | .423       | .413      | .543       | .490      |
|                                       | +LGC     | .471        | .424       | .424      | .577       | .573      |

Table 2: **Main results on MID-Bench.** Clear evaluates standard action accuracy on the Clear split. Base/Hard Ambig are evaluated under two scoring rules: *Match-Only*, which credits any GT-matching action regardless of asking, and *Ask+Match*, which additionally requires `call_user` before a correct final action. Train-free clarification cascades substantially improve Ask+Match while largely preserving Clear performance. Bold marks the per-base top in each column.

the most damage. All gates beat baseline on every Match-Only cell, and every cascade row continues to beat baseline under Ask+Match, so the cascade advantage is not a scoring-rule artifact; tightening to Ask+Match widens these gaps further still.

**Ask+Match exposes the baseline collapse and the cascade lift.** Tightening the rule to require `call_user` drops every baseline to Ask+Match = 0: the Match-Only – Ask+Match gap on each baseline row is Act-without-Asking measured as a score, and shrinks  $\approx 4\times$  from *Base* to *Hard* but stays above zero. Against the 0 floor, cascades lift Ask+Match to .134–.536 on *Base* and .150–.563 on *Hard*, with one outlier: SGC drops sharply on UI-Venus-Navi-7B (.051 / .037), while PGC/LGC on the same base reach .435/.536 on *Base* and .507/.563 on *Hard*. On *Hard*, the best cascade matches or exceeds the best cascade on *Base* for three of four bases (GUI-Owl-1.5-4B: .460 vs. .478), so the lift persists exactly where baselines collapse. The three gates are not interchangeable: PGC lags on the 4B bases, SGC performs well on GUI-Owl-1.5-4B, and LGC attains the strongest

aggregate results on Qwen3-VL-4B and UI-Venus-Navi-7B. The logit gate consistently outperforms the prompted gate on the 4B models, while all three gates converge on UI-TARS-1.5-7B.

**LoRA SFT helps and combines effectively with our cascades.** LoRA SFT alone lifts Qwen3-VL-4B’s Ask+Match from 0 to .333 on *Base* and .230 on *Hard*, while leaving standard Clear performance intact (.493 vs. .468 for the frozen Qwen3-VL-4B baseline), indicating that MID-Bench provides a learnable supervision signal for ambiguity-aware acting. The gains nevertheless trail our train-free Qwen3-VL-4B+LGC pipeline on *Hard* ambiguity (LoRA .230 vs. Qwen3-VL-4B+LGC .423), suggesting that supervised adaptation alone is insufficient under stricter ambiguity settings. Composing the two yields the strongest overall results: LoRA+LGC reaches Ask+Match *Hard* **.573**, the top result in the table, while LoRA+SGC reaches .490. This suggests that LoRA improves ambiguity sensitivity, while explicit gating remains important under harder ambiguity settings and largely preserves Clear performance. We additionally eval-

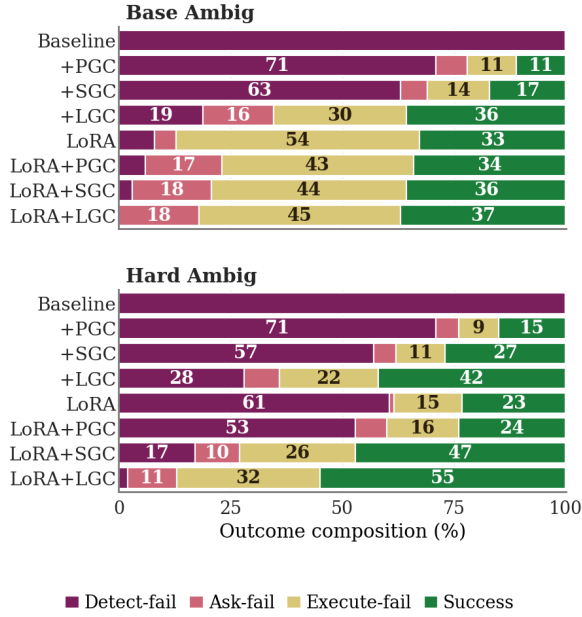


Figure 6: **Per-stage outcome decomposition.** Each bar sums to 100% over four outcomes: *Detect-fail* (the gate does not fire), *Ask-fail* (the simulator rejects the clarification question), *Execute-fail* (clarification succeeds but the final action is incorrect), and *Success* (successful clarification and execution).

uate a closed-source reference point using GPT-5.4 (App. O), which exhibits stronger native clarification behavior than the frozen open-weight baselines but still benefits from explicit clarification gating.

## 7 Further Analysis

### 7.1 Where each method fails: detection, asking, or final action

We decompose each ambiguous step on Qwen3-VL-4B into four outcomes: **Detect-fail** (the gate does not trigger clarification), **Ask-fail** (clarification is triggered but rejected by the simulator), **Execute-fail** (clarification succeeds but the final action is incorrect), and **Success** (successful clarification and execution) (Fig. 6).

The dominant failure mode differs sharply across methods. On the frozen base, both PGC and SGC fail primarily at detection, with Detect-fail accounting for 57–71% of all ambiguous steps across *Base/Hard*. In contrast, LGC substantially reduces Detect-fail (to 19%/28% on *Base/Hard*), shifting failures toward post-clarification execution and correspondingly improving success rates. LoRA changes the profile more fundamentally: on *Base* Ambig, most residual failures become Execute-fail rather than Detect-fail, suggesting that ambiguity

| Method            | Referential | Under-specified | Preference  | Contextual  |
|-------------------|-------------|-----------------|-------------|-------------|
| <i>Base Ambig</i> |             |                 |             |             |
| PGC               | .252        | .123            | .094        | .142        |
| SGC               | .117        | .189            | .177        | .179        |
| LGC               | .372        | <b>.376</b>     | <b>.364</b> | <b>.423</b> |
| LoRA              | <b>.570</b> | .227            | .296        | .367        |
| <i>Hard Ambig</i> |             |                 |             |             |
| PGC               | .056        | .119            | .138        | .283        |
| SGC               | .278        | .233            | .483        | .300        |
| LGC               | <b>.556</b> | <b>.383</b>     | <b>.586</b> | <b>.433</b> |
| LoRA              | .167        | .218            | .207        | .300        |

Table 3: **Ask+Match breakdown by ambiguity category.** Performance varies substantially across ambiguity types.

detection is largely recovered. Under *Hard* Ambig, however, additional gating remains important for suppressing residual Detect-fail and achieving the strongest overall performance.

### 7.2 Performance across ambiguity categories

Tab. 3 reveals substantial variation across ambiguity categories. LoRA performs best on *Referential* ambiguity in the *Base* split (.570), suggesting that lightweight supervised adaptation is effective when ambiguity can be resolved through short-horizon reference grounding. However, LoRA degrades considerably on the stricter *Hard* split, where action-critical text is explicitly removed. In contrast, LGC achieves the strongest overall robustness, topping the remaining seven cells and peaking at .586 on *Preference Hard*, indicating that explicit clarification gating generalizes more consistently across ambiguity types.

*Underspecified* ambiguity remains challenging: it is LoRA’s weakest *Base* category (.227) and LGC’s weakest *Hard* category (.383), where the typed value is verifiably masked. Unlike referential or contextual ambiguity, these cases often lack sufficient constraints to infer the next action without acquiring genuinely new user information. This suggests that underspecified cases expose a more fundamental limitation of current GUI agents beyond simple ambiguity recognition or local reference resolution.

## 8 Conclusion

We introduce MID-Bench, a benchmark for evaluating ambiguity-aware acting in mobile GUI agents and the Act-without-Asking failure mode, where agents continue acting under ambiguity instead of

requesting clarification. Across four open-weight agents, we find a consistent recognition–action gap, while train-free clarification gates substantially improve ambiguous-task performance with minimal loss on standard GUI actions.

## Limitations

MID-Bench focuses on ambiguity in mobile GUI agents under a single-round clarification protocol centered on the action-blocking ambiguity point. Extending to Web/desktop agents, longer-horizon interactions with accumulated dialogue history, and broader agent families remains future work.

**Potential risks.** Ambiguity-aware GUI agents may still produce unintended actions when clarification fails or user intent remains underspecified. In addition, MID-Bench inherits screenshots and interaction trajectories from existing mobile GUI benchmarks, which may involve redistribution, licensing, or privacy considerations.

## References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pages 475–484.
- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. 2025. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*.
- 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.
- Ziming Cheng, Zhiyuan Huang, Juntong Pan, Zhaohui Hou, and Mingjie Zhan. 2025. Navi-Plus: Managing ambiguous GUI navigation tasks with follow-up questions. *arXiv preprint arXiv:2503.24180*.
- Jeremy Cole, Michael Zhang, Dan Gillick, Julian Eisen-schlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 530–543.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.
- Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. 2025. UI-Venus technical report: Building high-performance UI agents with RFT. *arXiv preprint arXiv:2508.10833*.
- Chao Hao, Shuai Wang, and Kaiwen Zhou. 2025. Uncertainty-aware GUI agent: Adaptive perception through component recommendation and human-in-the-loop refinement. *arXiv preprint arXiv:2508.04025*.
- Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.
- Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2024. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Liang Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.
- Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. 2025. MobileWorld: Benchmarking autonomous mobile agents in agent-user interactive and MCP-augmented environments. *arXiv preprint arXiv:2512.19432*.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. CLAM: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*.
- Wei Li, William Bishop, Alice Li, Chris Rawles, Fola-iyi Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on UI control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.
- Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024. AutoGLM: Autonomous foundation agents for GUIs. *arXiv preprint arXiv:2411.00820*.
- Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025. GUI-Odyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.
- Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 5783–5797.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*.
- Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746.
- Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. 2025. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pages 406–441.
- Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. AndroidInTheWild: A large-scale dataset for Android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728.
- Aymeric Roucher and Amir Mahla. 2025. ScreenSuite: The most comprehensive benchmarking suite for GUI agents. <https://github.com/huggingface/screensuite>.
- Jiazheng Sun, Mingxuan Li, Yingying Zhang, Jiayang Niu, Yachen Wu, Ruihan Jin, Shuyu Lei, Pengrong Tan, Zongyu Zhang, Ruoyi Wang, et al. 2026. AmbiBench: Benchmarking mobile GUI agents beyond one-shot instructions in the wild. *arXiv preprint arXiv:2602.11750*.
- Jiazheng Sun, Ruimeng Yang, Xu Han, Jiayang Niu, Mingxuan Li, Te Yang, Yongyong Lu, and Xin Peng. 2025. Robust, observable, and evolvable agentic systems engineering: A principled framework validated via the Fairy GUI agent. *arXiv preprint arXiv:2509.20729*.
- Manan Suri, Puneet Mathur, Nedim Lipka, Franck Deroncourt, Ryan A Rossi, and Dinesh Manocha. 2025. Structured uncertainty guided clarification for LLM agents. *arXiv preprint arXiv:2511.08798*.
- Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442.
- Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. 2021. AndroidEnv: A reinforcement learning platform for Android. *arXiv preprint arXiv:2105.13231*.
- 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*.
- Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jentsse Huang, Wenxiang Jiao, and Michael R Lyu. 2025. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21784–21795.
- Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. 2025. OS-ATLAS: Foundation action model for generalist GUI agents. In *International Conference on Learning Representations*, volume 2025, pages 5090–5108.
- Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. 2026. Mobile-Agent-v3.5: Multi-platform fundamental GUI agents. *arXiv preprint arXiv:2602.16855*.
- Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. AppAgent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.
- Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024a. Android in the zoo: Chain-of-action-thought for GUI agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031.

- Michael JQ Zhang and Eunsol Choi. 2025. Clarify when necessary: Resolving ambiguity through interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5526–5543.
- Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024b. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10766.
- Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. 2025. MAI-UI technical report: Real-world centric foundation GUI agents. *arXiv preprint arXiv:2512.22047*.

## A Prompt Templates

We list the load-bearing prompts used in MID-Bench. Each base agent uses its official cookbook system prompt verbatim; we summarize the cookbook structure below and list the `call_user` hook injected on top.

### A.1 GUI-agent system prompts

**Qwen3-VL.** Copied verbatim from the Qwen3-VL `mobile_agent.ipynb` cookbook.<sup>1</sup> Coordinate system: 999×999 grid. Action space: the `mobile_use` tool exposes 8 actions (`click`, `long_press`, `swipe`, `type`, `answer`, `system_button`, `wait`, `terminate`). Response format: `Thought:`, `Action:`, and a single `<tool_call>`-wrapped JSON object.

**GUI-Owl-1.5.** Copied from the Mobile-Agent-v3.5 `end2end_usage_mobile` cookbook prompt, built via `NousFnCallPrompt` (`qwen-agent`).<sup>2</sup> Coordinate system: 999×999 grid, same as Qwen3-VL. Action space: the standard mobile action enum (same set as Qwen3-VL). Response format: same JSON tool-call schema as Qwen3-VL, with a per-step task-progress field appended to the user message.

**UI-TARS-1.5.** The official `MOBILE_USE DOUBAO` template.<sup>3</sup> Coordinate system: absolute pixel coordinates with `<point>x y</point>` notation, scaled via `smart_resize`. Action space: 10 actions (`click`, `long_press`, `type`, `scroll`, `open_app`, `drag`, `press_home`, `press_back`, `wait`, `finished`). Response format: `Thought:/Action:` plain-text format.

**UI-Venus-Navi.** Copied from the official `USER_PROMPT` template in the UI-Venus navigation framework.<sup>4</sup> Coordinate system: normalized float coordinates (e.g. `box=(x,y)`). Action space: 13 actions including a native `CallUser` (`Click`, `Drag`, `Scroll`, `Type`, `Launch`, `Wait`, `Finished`, `CallUser`, `LongPress`, `PressBack`, `PressHome`, `PressEnter`, `PressRecent`). Response format: `<think>/<action>/<conclusion>` tags.

<sup>1</sup>[https://github.com/QwenLM/Qwen3-VL/blob/main/cookbooks/mobile\\_agent.ipynb](https://github.com/QwenLM/Qwen3-VL/blob/main/cookbooks/mobile_agent.ipynb)

<sup>2</sup>[https://github.com/X-PLUG/MobileAgent/blob/main/Mobile-Agent-v3.5/cookbook/end2end\\_usage\\_mobile.ipynb](https://github.com/X-PLUG/MobileAgent/blob/main/Mobile-Agent-v3.5/cookbook/end2end_usage_mobile.ipynb)

<sup>3</sup>[https://github.com/bytedance/UI-TARS/blob/main/codes/ui\\_tars/prompt.py](https://github.com/bytedance/UI-TARS/blob/main/codes/ui_tars/prompt.py)

<sup>4</sup><https://github.com/inclusionAI/UI-Venus/blob/main/models/navigation/utils.py>

### A.2 `call_user` injection

UI-Venus-Navi-7B already exposes `CallUser` in its cookbook action list, so no injection is needed. For the other three bases, we add `call_user` directly to the action space with the description:

*“Call the user when the situation is ambiguous, risky, or you are unsure how to proceed.”*

### A.3 Ambiguity classifier (3-way)

For the recognition column of Tab. 1, each base classifies (screen, instruction) without producing an action:

*Look at this phone screen and the instruction “{instruction}”. Classify exactly one of: CLEAR (next action is uniquely determined), AMBIGUOUS (multiple plausible actions; agent should ask), NO\_TASK (no real task on this screen). Reply with only one of the three labels.*

### A.4 Strict user simulator

The strict simulator (§4.2) answers only questions whose answer is uniquely determined by the original (unmasked) goal:

*You simulate the user, hidden goal “{original\_instruction}”. The agent asked: “{question}”.*

**Rules.** (1) Reply with only the piece of information the question asks for. (2) If the question is vague (no specific info named), reply: “What specifically do you need to know?” (3) Keep replies to 2–5 words.

**Examples.** “Which app?” → “Gmail.” “What text?” → “Killing.” “Should I use this one?” → “yes.” “Can you tell me more?” → “What specifically do you need to know?”

### A.5 PGC verbalized gate

PGC (§5) wraps the base with a separate verbalized gate prompt, run before action generation:

*Look at this phone screen.  
Task: “{instruction}” (plus optional action history).*

*Do you have enough information to confidently perform a specific action?*

**Examples.** “Open Settings and turn on Wi-Fi” + Settings screen → YES. “Send a message” + Home screen → NO (missing who/what).  
*Reply YES or NO only.*

A NO fires a question generator: “You don’t have enough information. Ask ONE clear, specific question. Reply with ONLY the question.”

## B Baseline Action-Type Distribution under Ambiguity

Fig. 1c reports that `call_user` accounts for only 0.1% of baseline emissions on the ambiguous pool

| action_type   | Qwen3-VL-4B | GUI-Owl-1.5-4B | UI-TARS-1.5-7B | UI-Venus-Navi-7B |
|---------------|-------------|----------------|----------------|------------------|
| click         | 365         | 536            | 464            | 465              |
| type          | 467         | 320            | 317            | 374              |
| system_button | 43          | 33             | 85             | 51               |
| swipe         | 18          | 9              | 23             | 6                |
| terminate     | 6           | 0              | 3              | 2                |
| wait          | 0           | 1              | 6              | 1                |
| long_press    | 0           | 1              | 0              | 0                |
| call_user     | 1           | 0              | 2              | 1                |

Table 4: **Baseline action-type distribution on the  $n=900$  ambiguous pool** (Base 600+ Hard 300). Counts per base; columns sum to 900. call\_user pools to  $4/3600 = 0.111\%$  across the four bases, supporting the 0.1% figure in Fig. 1c.

( $n=900$ ). Tab. 4 reports the full distribution: across the four open-weight bases, GUI manipulation actions (click, type, swipe, system\_button) absorb  $> 98.5\%$  of emissions per base (98.8–99.8%; 99.3% pooled), while call\_user pools to  $4/3600=0.111\%$ .

## C Dataset Construction Prompts

The construction pipeline (§4.1) uses the following prompts, in pipeline order. The pivot-step prompt sees both the original and the rewritten instruction together with the trajectory’s action sequence.

### Four-type instruction rewriting.

Convert this clear mobile task instruction into an ambiguous version.  
Ambiguity type: {amb\_type} — {description}.  
Example. Clear: “{example\_orig}”. Ambiguous: “{example\_ambig}”.  
Now convert. Clear: “{instruction}”.  
Reply **ONLY** with JSON, no markdown:  
{"ambiguous\_instruction": "...",  
"missing\_info": ["info1", "info2"]}.

The four amb\_type values and their descriptions: *Referential* (replace specific names with vague references), *Underspecified* (remove key parameters needed to complete the task), *Preference-dependent* (replace objective criteria with subjective preferences), *Contextual* (assume prior context that doesn’t exist); each comes with one fixed clear/ambiguous example pair.

### Pivot-step identification.

You are analyzing a mobile phone task to determine when clarification is needed.  
**ORIGINAL** clear instruction: “{original}”. **AMBIGUOUS** version (given to the agent): “{ambiguous}”.  
The episode has these steps: {steps} (one line per step: action type, typed text, coordinate).  
At which step does the agent **FIRST** need the missing information to decide what to do?  
**Rules.** If step 0 is “open app” and the app name

is clear from context, that step does **NOT** need clarification; if the ambiguous instruction says “that app” with no context, it **DOES**. “type” actions almost always need clarification if the text content is part of the missing info. “click” on a specific item needs clarification if the item choice depends on the missing info. Generic navigation (back, home, scroll) usually does **NOT** need clarification.

Reply with **ONLY** a JSON: {"pivot\_step": N, "reason": "brief explanation"}. If **ALL** steps can be reasonably performed without the missing info, reply {"pivot\_step": -1, "reason": "no clarification needed"}.

### Hard typed-text masking.

Original instruction: “{orig}”. The agent’s step types this exact text: “{typed}”.

**Rules.** (1) Remove all mention of “{typed}”; the rewrite **MUST NOT** contain any token of length  $\geq 4$  from it. (2) Keep task structure (which app, general action). (3) Use a vague reference (e.g. “type my email”, “enter the address”). (4) Read naturally, like a real user being non-specific. Output only the rewritten instruction, one line.

We auto-verify the length- $\geq 4$  no-leak constraint and discard rewrites that violate it.

## D Internal Validation Study

We validate the released  $n=900$  Ambig pool in two stages. **Stage 1.** (Q1) whether the rewritten instruction is genuinely ambiguous given the screenshot, applied to all  $n=900$  items; and (Q2) which ambiguity category dominates, one of five categories (referential, underspecified, preference, contextual, other), applied to the  $n=830$  items confirmed ambiguous by Q1. **Stage 2.** The underspecified subset from Q2 is further sub-classified into three slot types (value, target, mixed), yielding a final 7-way taxonomy reported in Fig. 4. Annotation was performed by the authors; no external annotators were recruited and no compensation was issued. Annotators see the screenshot, the rewritten instruction, and a translucent overlay of the GT next action (red marker for click, text badge for type/open\_app). The annotator-facing prompt is:

*You are auditing a sample in a mobile-GUI “ambiguous instruction” benchmark. REWRITTEN\_INSTRUCTION: “{ambig}”. ORIGINAL\_INSTRUCTION (context only, hidden at runtime): “{orig}”. GT\_ACTION: {action}. Q1. Is the rewritten instruction actually ambiguous on this screen? yes / no / unsure. Q2. Most prominent ambiguity type: referential / underspecified / preference / contextual / not\_ambig (iff Q1=no) / other. Reply: Q1: ..., Q2: ..., REASON: <one sentence>.*

Q1 confirms ambiguity on 830/900 items (92.2%); we retain all released examples for evaluation to preserve the original benchmark distribution. Q2 is then applied to these 830 confirmed-ambiguous items, and the resulting category mix is summarized in Fig. 4.

**Underspecified subdivision.** Items labelled underspecified in Q2 receive a second-stage pass that subdivides the missing slot into *value*, *target*, or *mixed*, producing the Underspec breakdown in Fig. 4. Annotators see the same screenshot and overlay as Q1/Q2 and are given the following prompt:

*Sub-classify which kind of slot is missing for this underspecified sample. REWRITTEN\_INSTRUCTION: “{ambig}”. GT\_ACTION: {action}. Options. value: action verb is clear (type/search/set/enter) but the literal value is missing. target: the UI element/app/file/item to act on is missing. mixed: both slots are missing or the case doesn’t fit cleanly. Reply: UNDER\_TYPE: value|target|mixed, REASON: <one sentence>.*

Three authors independently annotated a subset of the released examples following the protocol above.

We use this as a qualitative sanity check on dataset quality rather than a large-scale human evaluation.

| Setting     | Base ( $n=600$ ) |       |          | Hard ( $n=300$ ) |       |          |
|-------------|------------------|-------|----------|------------------|-------|----------|
|             | Oracle           | Local | $\Delta$ | Oracle           | Local | $\Delta$ |
| Qwen3-VL-4B | .000             | .000  | .000     | .000             | .000  | .000     |
| +PGC        | .134             | .127  | −.007    | .150             | .147  | −.003    |
| +SGC        | .171             | .189  | +.018    | .273             | .250  | −.023    |
| +LGC        | .407             | .445  | +.038    | .423             | .383  | −.040    |

Table 5: **Local simulator swap.** Ask+Match on the  $n=900$  ambiguous-goal pool when replacing the GPT-4o strict simulator with the agent’s own Qwen3-VL-4B model. *Oracle* uses the GPT-4o strict simulator from §4.2 (reproduced from the Qwen3-VL-4B rows of Tab. 2); *Local* uses the same Qwen3-VL-4B weights under the strict-persona prompt, with the original gate thresholds  $\tau^*$  transferred unchanged. Performance shifts remain within  $\pm .040$  across all settings, with the largest change (.040) on LGC Hard.

## E Local User Simulator Swap

We used GPT-4o as the strict simulator (§4.2) in our main results. To check whether it can be replaced at lower cost, we swap it with the same Qwen3-VL-4B model that produces actions, otherwise following the original simulator setup.

Ask+Match shifts remain within  $\pm .040$  in every cell. The three gate signals—PGC verbalization, SGC entropy, and LGC logit—are simulator-independent by construction, since each fires before any simulator call. The swap therefore only affects post-ask final-action quality. A 4B general-purpose VLM can consequently drive both the agent and the strict simulator with shifts within  $\pm .040$ , removing the closed-API dependency (Tab. 5).

## F Simulator Persona Ablation

The strict simulator (§4.2) deflects underspecified questions. To bound the effect of this persona choice, we rerun the LGC cascades with a fully cooperative persona that always answers substantively. Every gate decision is unchanged (ask rates are identical on all four bases, since all gate signals fire before any simulator call), and *Hard* Ask+Match moves by at most .070 (Tab. 6). The shift is upward on three of four bases and slightly downward on one (−.027), which we attribute at least in part to simulator randomness; the strict persona we report with is therefore the conservative choice.

## G Multi-Round Clarification Extension

Our protocol caps clarification at a single round to isolate the upstream gate decision from question repair and answer aggregation. To verify that the cap is not binding in practice, we extend the protocol on *Hard* in the ask-enabled configuration

| Base (w/ LGC)    | <i>Hard</i> Ask+Match $\Delta$ (coop. − strict) |
|------------------|-------------------------------------------------|
| Qwen3-VL-4B      | −.027                                           |
| GUI-Owl-1.5-4B   | +.070                                           |
| UI-TARS-1.5-7B   | +.043                                           |
| UI-Venus-Navi-7B | +.070                                           |

Table 6: **Simulator persona ablation.** *Hard* Ask+Match shift when the strict simulator is replaced with a fully cooperative one that always answers substantively. Gate decisions are unchanged (identical ask rates on all four bases); only the post-ask tail moves.

| Base             | Ask+Match by budget |      |      | Rounds used |     |
|------------------|---------------------|------|------|-------------|-----|
|                  | 1                   | 2    | 3    | 2nd         | 3rd |
| Qwen3-VL-4B      | .553                | .553 | .553 | 1           | 0   |
| GUI-Owl-1.5-4B   | .520                | .520 | .520 | 1           | 0   |
| UI-TARS-1.5-7B   | .420                | .420 | .420 | 0           | 0   |
| UI-Venus-Navi-7B | .510                | .510 | .510 | 2           | 0   |

Table 7: **Multi-round extension on *Hard*.** Ask-enabled configuration (a first question is guaranteed) with the round budget raised to 2 and 3; a separate run from the always-ask rows of Tab. 10, so absolute values differ by simulator randomness and only the across-budget comparison is of interest. Ask+Match is unchanged to three decimals on every base; the extra rounds are almost never used (counts out of 300 interactions per base).

(a first question is guaranteed), raising the round budget to 2 and then 3, so a deflected first question can be followed by a second, more specific one rather than being scored 0. Ask+Match is unchanged to three decimals on every base at every budget (Tab. 7), and agents almost never use the extra rounds: a second round is taken in at most 2 of 300 interactions per base, and a third in none. The cap therefore does not distort measurement; the binding failure remains the upstream ask-or-act decision. Dialogues that withhold several independent pieces of information remain outside this measured scope.

## H LoRA Data Recipe and Training Setup

**Data construction.** The training set contains 4,488 examples built from 1,496 (screenshot, GT action) triples drawn from the AndroidControl training split. Each triple yields three examples on the same screen with the same GT action: (i) the original clear instruction  $\rightarrow$  GT action; (ii) the ambiguous rewrite  $\rightarrow$  call\_user with a GPT-4o-grounded clarification question; and (iii) the ambiguous rewrite plus the strict simulator’s answer  $\rightarrow$  GT action. The base policy, the ask decision, and the post-answer action are thus trained jointly; the answer is appended to the instruction as a single turn rather than as a multi-turn dialogue.

**Training setup.** LoRA adapters are attached to Qwen3-VL-4B with rank 16,  $\alpha=32$ , and dropout 0.05. We train for 2 epochs at  $LR=2\times 10^{-4}$  on  $2\times$  NVIDIA RTX 3090 GPUs, totaling approximately 2 GPU-hours.

| Setting                 | Base |      | Hard |      |
|-------------------------|------|------|------|------|
|                         | AC   | Ody  | AC   | Ody  |
| Qwen3-VL-4B + LGC       | .429 | .362 | .433 | .413 |
| Qwen3-VL-4B(LoRA)       | .444 | .111 | .153 | .307 |
| Qwen3-VL-4B(LoRA) + LGC | .565 | .141 | .633 | .513 |

Table 8: **Cross-dataset split of Ask+Match** (Qwen3-VL-4B cells of Tab. 2; baseline rows = 0 omitted). AC: AndroidControl ( $n_{\text{Base}}=400$ ,  $n_{\text{Hard}}=150$ ). Ody: GUI-Odyssey ( $n_{\text{Base}}=200$ ,  $n_{\text{Hard}}=150$ ). LGC transfers to Ody on both splits; LoRA shows unstable cross-dataset transfer when applied alone, with the AC–Ody gap flipping sign across splits.

## I Cross-Dataset Replication on GUI-Odyssey

We split Tab. 2’s Qwen3-VL-4B cells by source dataset to check whether train-free gains generalize across data distributions (Tab. 8). Frozen Qwen3-VL-4B+LGC performs comparably on AC and Ody on *Hard* and transfers to Ody with a moderate drop on *Base* (.43→.36), while LoRA SFT, trained only on AC, shows unstable cross-dataset transfer, with the AC–Ody gap flipping sign between Base (Ody much lower) and Hard (Ody higher). LGC thus contributes dataset-agnostic generality, whereas LoRA contributes within-distribution quality.

## J F1 Decomposition

Should-ask F1 is the harmonic mean of recall (call\_user fire rate on Ambig) and precision (TP/(TP+FP), where FP is a call\_user fire on a Clear task), treating Ambig as the positive class. Per-cell decomposition in Tab. 9.

| Cell              | Prec. | Recall | F1          |
|-------------------|-------|--------|-------------|
| Qwen3-VL-4B       | 1.000 | .001   | .002        |
| +PGC              | .671  | .290   | .405        |
| +SGC              | .828  | .390   | .530        |
| +LGC              | .767  | .781   | <b>.774</b> |
| GUI-Owl-1.5-4B    | –     | .000   | .000        |
| +PGC              | .696  | .331   | .449        |
| +SGC              | .792  | .773   | .782        |
| +LGC              | .793  | .793   | <b>.793</b> |
| UI-TARS-1.5-7B    | 1.000 | .002   | .004        |
| +PGC              | .775  | .904   | <b>.834</b> |
| +SGC              | .761  | .910   | .829        |
| +LGC              | .775  | .884   | .826        |
| UI-Venus-Navi-7B  | 1.000 | .001   | .002        |
| +PGC              | .757  | .888   | .817        |
| +SGC              | .840  | .076   | .139        |
| +LGC              | .758  | .999   | <b>.862</b> |
| Qwen3-VL-4B(LoRA) | .991  | .749   | .853        |

Table 9: **Per-cell should-ask F1 decomposition** for the cells of Tab. 2 ( $n_{\text{amb}}=900$ ,  $n_{\text{clear}}=300$ ). PGC precision lags on the 4B bases; LGC reaches the highest F1 on Qwen3-VL-4B/GUI-Owl-1.5-4B via high recall with competitive precision. On UI-TARS-1.5-7B all three gates converge ( $\pm.008$  F1). SGC on UI-Venus-Navi-7B collapses on recall. Qwen3-VL-4B(LoRA) baseline reaches F1= .853 via supervised native asking.

## K Oracle-Gate Ceiling

To separate information access from gate quality, we report an oracle-gate control computed from the existing traces: on the all-ambiguous pools, a perfect should-ask detector fires on every step, so the oracle gate coincides with always-ask. This yields the information ceiling of the clarification pipeline. The ceiling is .48–.61 rather than 1.0 because question quality and re-acting on the simulator’s answer still limit performance. Tab. 10 compares this ceiling with the best cascade per base on *Hard* Ask+Match: the train-free cascades recover 72–99% of the ceiling, and LoRA+LGC recovers 99%. *Base* behaves the same, with oracle ceilings of .44–.56 on the four frozen bases.

| Base               | Oracle (= always-ask) | Best cascade    | Recovered |
|--------------------|-----------------------|-----------------|-----------|
| Qwen3-VL-4B        | .590                  | .423 (LGC)      | 72%       |
| GUI-Owl-1.5-4B     | .607                  | .460 (SGC)      | 76%       |
| UI-TARS-1.5-7B     | .480                  | .440 (LGC)      | 92%       |
| UI-Venus-Navi-7B   | .567                  | .563 (LGC)      | 99%       |
| Qwen3-VL-4B (LoRA) | .577                  | .573 (LoRA+LGC) | 99%       |

Table 10: **Oracle-gate ceiling on *Hard Ask+Match***. The oracle gate fires on every ambiguous step, which coincides with always-ask on the all-ambiguous pool, and gives the information ceiling of the ask→answer→re-act pipeline. Best cascade per base from Tab. 2; Recovered = best cascade / oracle.

## L Sampling-Gate Cascade Implementation

SGC (§5) fires when  $k=5$  stochastic decodes of the baseline action disagree under a coarse action-equivalence relation. We sample using each base agent’s native decoding pipeline (temperature  $T=0.7$ , top- $p=1.0$ ), while reusing the same action parser as the baseline pass so sampled outputs remain directly comparable to the original action prediction. Sampled action signatures are cached on disk, yielding an amortized inference cost of  $k+1$  baseline passes across reruns.

Each parsed action is mapped to a string signature (Tab. 11). click coordinates are bucketed onto a 50-unit grid in each base’s native coordinate system; typed strings are case-folded, whitespace-stripped, and truncated to 40 characters; press preserves the button identity; swipe collapses to a single class regardless of direction; and all remaining actions collapse to their action-type string. Unparseable samples are discarded. These heuristics are intentionally coarse and are designed to capture semantically distinct action proposals rather than exact low-level equivalence.

Letting  $n_i$  denote the count of signature class  $i$  and  $p_i = n_i / \sum_j n_j$  the corresponding empirical class probabilities, we compute Shannon entropy  $H = -\sum_i p_i \log p_i$  in nats. The entropy ranges from 0 when all samples collapse to a single signature class to  $\log 5 \approx 1.609$  under a uniform distribution over all five samples. The gate fires iff  $H > \tau_{\text{SGC}}$ ; if no sample parses (no signal), the gate does not fire and the baseline outcome is scored. We use  $\tau_{\text{SGC}}=0$  uniformly across all bases (App. N), which is equivalent to triggering whenever at least one of the five sampled actions disagrees with the others. At this setting SGC reduces to a disagreement detector; we keep the entropy formulation because it generalizes directly to nonzero thresholds.

When the gate fires, the wrapper suppresses the

| Action type | Signature                        |
|-------------|----------------------------------|
| click       | click:{x//50},{y//50}            |
| type        | type:{text.strip().lower()[:40]} |
| press       | press:{button}                   |
| swipe       | swipe                            |
| call_user   | call_user                        |
| other       | {action_type}                    |

Table 11: **SGC action-equivalence signatures**. Two samples are grouped into the same class iff their signatures match exactly.

baseline action and enters the clarification stage (ask→answer→final action) shared with PGC and LGC.

## M Per-Gate Cost Model and Measured Latency

Expected per-step inference counts follow from each gate’s structure: LGC costs  $1+2\varphi$  passes, PGC  $2+2\varphi$ , and SGC  $1+k+2\varphi$ , where  $\varphi$  is the gate fire rate and the  $2\varphi$  term is the clarification tail (question + re-action) paid only on fired steps; SGC pays its  $k$  sampling passes on every step, before the gate decision. Tab. 12 reports per-step wall-clock latency measured on a single RTX 3090 Ti endpoint (30 steps per component; ranges span the four bases). The base pass alone takes 1.9–3.0 s, and the clarification tail of  $\approx 3.9$ –5.3 s is paid only on fired steps. The measurements match the cost model: SGC runs at  $\approx 2$ – $3\times$  the per-step cost of either other gate at matched  $\varphi$ , while LGC adds no gate overhead because it reuses the logits of the base pass.

## N Threshold $\tau$ : Selection and Sensitivity

Both tunable gates ( $\tau_{\text{SGC}}$ ,  $\tau_{\text{LGC}}$ ) are selected by maximizing should-ask F1 on a held-out validation split and transferred unchanged to evaluation. SGC uses a uniform threshold  $\tau=0$  across all bases, corresponding to the discrete operating point “at least one of five samples disagrees.”

| Method        | Added gate cost / step    | $\varphi=0.5$ | $\varphi=1.0$ |
|---------------|---------------------------|---------------|---------------|
| LGC           | none (reuses base logits) | 3.8–5.6       | 5.7–8.2       |
| PGC           | +0.45–0.95 s gate call    | 4.7–6.4       | 6.7–9.0       |
| SGC ( $k=5$ ) | +5.4–11.0 s sampling      | 9.2–16.6      | 11.2–19.2     |

Table 12: **Per-gate cost model and measured per-step latency (s).** Expected per-step inference counts: LGC  $1+2\varphi$ , PGC  $2+2\varphi$ , SGC  $1+k+2\varphi$  ( $\varphi$  = gate fire rate). Wall-clock seconds measured on a single-GPU endpoint; ranges span the four bases. SGC pays its  $k$  sampling passes on every step, before the gate decision.

LGC, in contrast, requires a base-specific threshold  $\tau^*$  because the gating statistic  $\log p$  (mean token log-probability) reflects model-dependent calibration of predictive uncertainty (Qwen3-VL-4B  $-0.035$ , GUI-Owl-1.5-4B  $-0.030$ , UI-TARS-1.5-7B  $-0.375$ , UI-Venus-Navi-7B  $-0.225$ ).

**Sensitivity sweep.** Tab. 13 sweeps  $\tau$  by  $\pm\{0.025, 0.05\}$  around  $\tau^*$  on four frozen bases, computed from the same released-pool traces as Tab. 2; the  $\Delta\tau=0$  row therefore coincides with the corresponding LGC cells of Tab. 2 and Tab. 9. All models are evaluated on the same combined AC+Ody pool ( $n_{\text{clear}}=300$ ,  $n_{\text{base}}=600$ ). Clear remains highly stable across the sweep ( $\leq .03$  variation for all bases), while Ask+Match on the *Base* pool increases monotonically with  $\tau$ . Should-ask F1 stays within .09 of its sweep maximum at  $\tau^*$  for every base, indicating that performance is robust to small perturbations of  $\tau$ . By contrast, SGC operates at a discrete boundary:  $\tau=0$  corresponds to the minimal disagreement regime (“ $\geq 1$  sample disagrees”), while any lower threshold degenerates into an always-fire policy.

| Base             | $\Delta\tau$ | Clear | Ask+Match<br>( <i>Base</i> ) | F1          |
|------------------|--------------|-------|------------------------------|-------------|
| Qwen3-VL-4B      | −0.050       | .474  | .035                         | .123        |
|                  | −0.025       | .468  | .161                         | .425        |
|                  | 0.000        | .489  | .407                         | .774        |
|                  | +0.025       | .490  | .509                         | <b>.857</b> |
|                  | +0.050       | .490  | .511                         | <b>.857</b> |
| GUI-Owl-1.5-4B   | −0.050       | .502  | .300                         | .587        |
|                  | −0.025       | .503  | .379                         | .696        |
|                  | 0.000        | .508  | .478                         | .793        |
|                  | +0.025       | .507  | .548                         | .850        |
|                  | +0.050       | .510  | .557                         | <b>.857</b> |
| UI-TARS-1.5-7B   | −0.050       | .405  | .251                         | .705        |
|                  | −0.025       | .400  | .299                         | .771        |
|                  | 0.000        | .403  | .357                         | .826        |
|                  | +0.025       | .405  | .394                         | .845        |
|                  | +0.050       | .408  | .421                         | <b>.854</b> |
| UI-Venus-Navi-7B | −0.050       | .525  | .513                         | .857        |
|                  | −0.025       | .536  | .527                         | .860        |
|                  | 0.000        | .542  | .536                         | <b>.862</b> |
|                  | +0.025       | .542  | .536                         | .860        |
|                  | +0.050       | .542  | .536                         | .858        |

Table 13: **LGC  $\tau$  sweep around  $\tau^*$ .** Clear, Ask+Match on *Base*, and should-ask F1 with  $\tau$  swept by  $\pm\{0.025, 0.05\}$  around  $\tau^*$  (F1-tuned on the held-out validation split), evaluated on MID-Bench. Bold marks the F1-maximizing row per base.

**Cross-dataset transfer.** To test whether the LGC threshold  $\tau^*$  is tied to a particular data distribution, we re-tune  $\tau_{\text{LGC}}$  on AndroidControl items alone and evaluate on GUI-Odyssey, and vice versa. The Ask+Match transfer penalty is small: 0 on Qwen3-VL-4B and GUI-Owl-1.5-4B (the transferred and self-tuned thresholds select identical operating points) and at most .037 on UI-TARS-1.5-7B and .013 on UI-Venus-Navi-7B. Threshold selection therefore does not require in-distribution tuning data.

| Method   | Base Ambig  |             | Hard Ambig  |             |
|----------|-------------|-------------|-------------|-------------|
|          | Match-Only  | Ask+Match   | Match-Only  | Ask+Match   |
| baseline | .385        | .145        | .370        | .360        |
| +PGC     | <b>.427</b> | <b>.280</b> | <b>.390</b> | <b>.380</b> |

Table 14: **GPT-5.4 reference panel on MID-Bench.** Match-Only and Ask+Match on Base and Hard ( $n=100$  per cell), self-initiated baseline vs. +PGC.

## O Closed-Model Reference: GPT-5.4

We additionally include a closed-source reference point using GPT-5.4 under the MID-Bench protocol. We use the Qwen3-VL cookbook prompt from App. A.1 with two minimal modifications: (1) replacing the screen-resolution placeholder with the actual screenshot resolution  $W \times H$  for each task, and (2) describing coordinates as absolute pixel positions rather than normalized coordinates. Returned coordinates are subsequently renormalized into the  $999 \times 999$  coordinate space expected by the scorer, leaving the remainder of the evaluation protocol unchanged.

GPT-5.4 exhibits substantially stronger native clarification behavior (Tab. 14) than the open-weight baselines of Tab. 2: unlike the frozen open models, the baseline already attains non-zero Ask+Match under the self-initiated protocol, indicating that the model can independently decide to ask for clarification under ambiguity. The behavior is markedly asymmetric across splits: on *Hard*, the Match-Only — Ask+Match gap nearly closes (.370 vs. .360), so almost every credited action follows a clarification; on *Base*, the gap stays wide (.385 vs. .145), indicating that GPT-5.4 reliably recognizes typed-text masking but largely acts without asking under softer ambiguity. Nevertheless, adding PGC still improves both Match-Only and Ask+Match across *Base* and *Hard*, suggesting that explicit clarification gating remains beneficial even for a strong closed-source model. At the same time, absolute performance remains below the strongest open-weight cascades in Tab. 2 under both scoring rules, indicating that clarification ability alone does not guarantee strong GUI control performance. We hypothesize that GPT-5.4’s relatively weak mobile-action grounding under the Qwen3-VL-style action interface limits downstream execution accuracy despite improved ambiguity handling.

## P Prior Ask-Trained Open Baseline: MAI-UI

Among prior GUI-agent approaches targeting ambiguous instructions (Cheng et al., 2025; Sun et al., 2025, 2026; Kong et al., 2025), we could not obtain the released data, code, or checkpoints required for a controlled step-level comparison. We therefore evaluate MAI-UI-8B (Zhou et al., 2025), a public checkpoint post-trained to ask clarification questions on ambiguous instructions, under the MID-Bench self-initiated protocol (native asking available, no cascade).

MAI-UI-8B is the only open-weight base we evaluate with nonzero Hard Ask+Match out of the box (.207, Tab. 15), while Base Ask+Match remains low (.065): ask-oriented post-training confers selective asking where the missing detail is verifiably masked, but misses softer ambiguity. Overall action accuracy remains below the Tab. 2 bases (Clear .395), so native asking alone does not close the gap to gated cascades.

## Q Dataset Examples

We visualize representative examples from both evaluation splits to illustrate the released benchmark schema. Each example contains: (1) the mobile screenshot, (2) the original fully specified instruction (**Clear Instruction**), provided only for reference and hidden from the agent, (3) the ambiguous instruction (**Ambiguous Instruction**) shown to the agent at evaluation time, and (4) the original ground-truth GUI action (**Ground-truth Action**) used for post-clarification evaluation. For click actions, a translucent red marker indicates the target coordinate on the screenshot. Figs. 7–8 show *Base* examples; Figs. 9–10 show *Hard* examples.

| Model     | Method   | Clear Split | Base Ambig |           | Hard Ambig |           |
|-----------|----------|-------------|------------|-----------|------------|-----------|
|           |          | Clear       | Match-Only | Ask+Match | Match-Only | Ask+Match |
| MAI-UI-8B | baseline | .395        | .384       | .065      | .257       | .207      |

Table 15: **MAI-UI-8B under the MID-Bench self-initiated protocol.** Same columns and pools as Tab. 2; no cascade is applied.

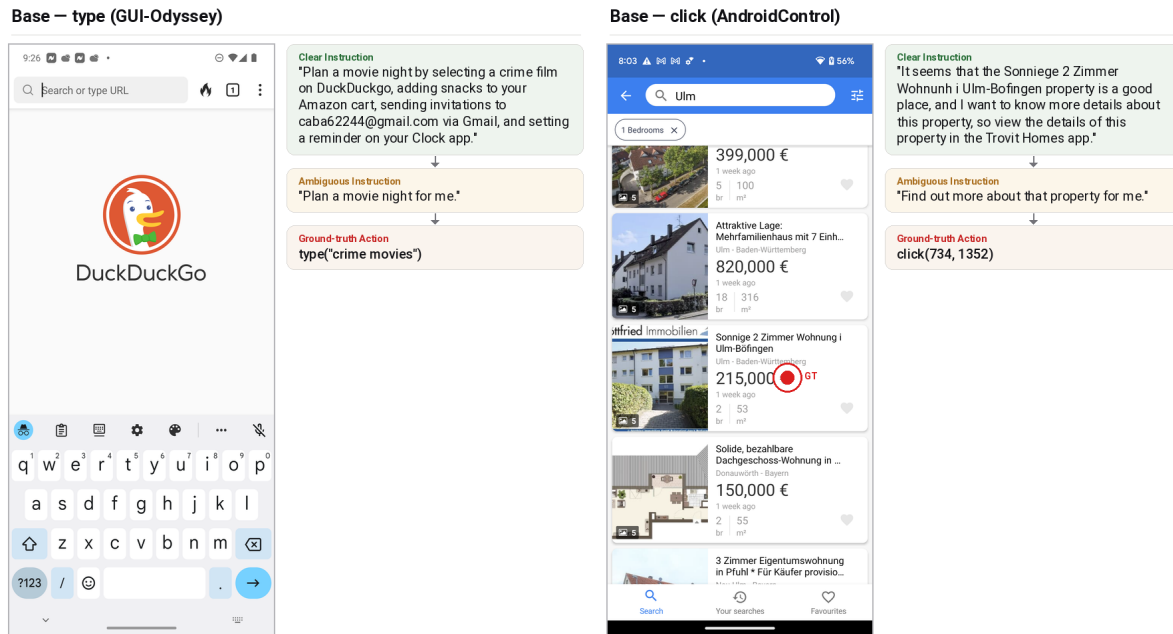


Figure 7: **Representative *Base* examples (1/2).**

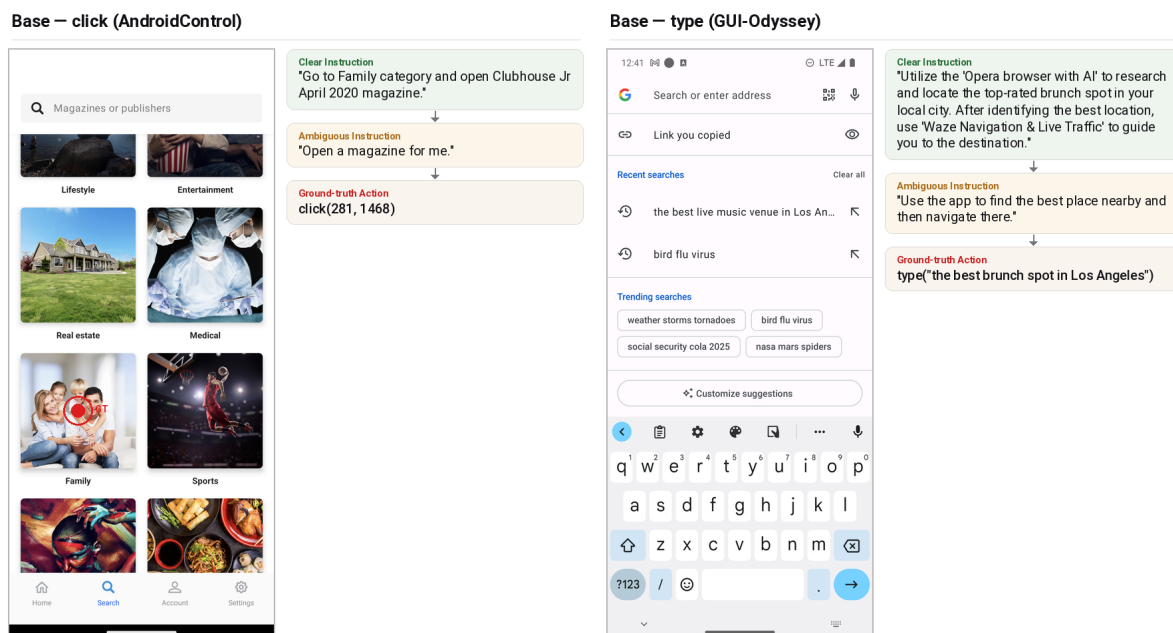
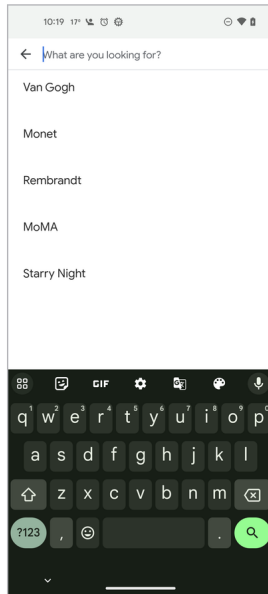


Figure 8: **Representative *Base* examples (2/2).**

#### Hard — type (AndroidControl)

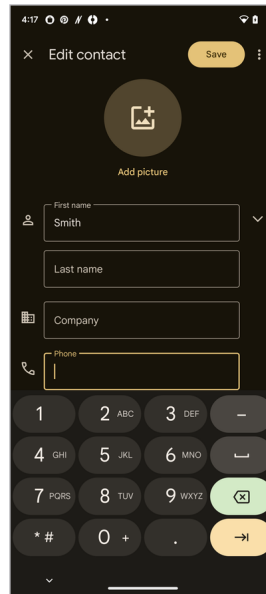


**Clear Instruction**  
"wassily kandinsky IV salon of Abstract artwork is what motivates me. On the Art & culture app, look for Wassily kandinsky."

**Ambiguous Instruction**  
"On the Art & culture app, search for that famous abstract artist I mentioned."

**Ground-truth Action**  
type("wassily kandinsky")

#### Hard — type (AndroidControl)



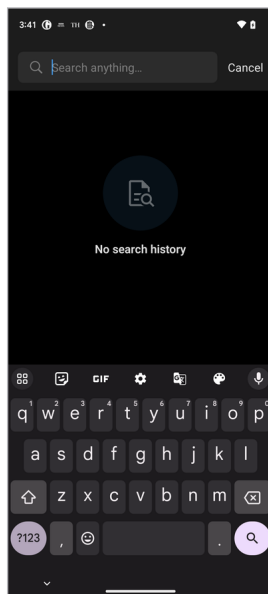
**Clear Instruction**  
"I need to change Smith, my neighbor's phone number. I must alter it to 9877655532 instead of 9877654532. Utilizing the contacts app, edit the contact"

**Ambiguous Instruction**  
"I need to change Smith, my neighbor's phone number to the correct one instead of the wrong one, using the contacts app, edit the contact"

**Ground-truth Action**  
type("9877655532")

Figure 9: Representative *Hard* examples (1/2).

#### Hard — type (AndroidControl)

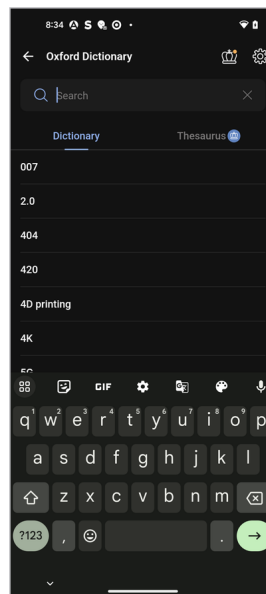


**Clear Instruction**  
"Open the smart news App search for covid 19 in the search bar"

**Ambiguous Instruction**  
"Open the smart news App and look up that topic in the search bar."

**Ground-truth Action**  
type("Covid 19")

#### Hard — type (AndroidControl)



**Clear Instruction**  
"In the Oxford dictionary app, search for the word hyper."

**Ambiguous Instruction**  
"In the Oxford dictionary app, search for a specific term."

**Ground-truth Action**  
type("hyper")

Figure 10: Representative *Hard* examples (2/2).