POLYAGENT: A MULTILINGUAL BENCHMARK FOR API-DRIVEN MOBILE DEVICE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Small Language Models (SLMs) increasingly enable mobile agents that translate natural-language intent into executable device and application actions. While recent benchmarks have advanced multilingual function calling, stateful tool use, and interactive mobile-agent evaluation, these capabilities remain only partially integrated. In particular, localized multilingual intent grounding for executable mobile APIs under dynamic environment feedback is insufficiently characterized. We introduce **PolyAgent**, a multilingual diagnostic benchmark for API-driven on-device mobile control. PolyAgent contains 5,400+ expert-validated queries across 6 typologically diverse languages and 47 real-world mobile APIs. A culturally grounded synthesis and native-expert validation pipeline targets localized intent and argument grounding rather than direct translation alone. Integrated with ToolSandbox, PolyAgent evaluates interactive execution through Trajectory Precision and Goal Success Rate (GSR), combining action-level fidelity with environment-state verification. Experiments across lightweight open-source models reveal substantial model- and language-dependent variation under dynamic execution: Qwen models exhibit pronounced degradation for several non-English languages, whereas Gemma models maintain comparatively stable cross-lingual performance while achieving strong overall accuracy. These results highlight the limitations of static or English-centric evaluation for multilingual mobile tool use. PolyAgent provides a unified diagnostic framework for evaluating localized, state-aware, and resource-constrained mobile agents.

1 INTRODUCTION

The rapid development of Small Language Models (SLMs)(Microsoft, 2024; Zhang et al., 2024; Liu et al., 2024) with increasingly capable reasoning and tool-use abilities is enabling a new paradigm of mobile interaction in which natural-language intent is translated into executable actions(Yao et al., 2023; Schick et al., 2023; Patil et al., 2024b). For on-device agents, this capability is particularly important: instead of relying exclusively on visual GUI manipulation(Zhang et al., 2023; Wang et al., 2024), an agent can invoke structured system and application APIs to modify device state, orchestrate applications, and complete tasks with low interaction overhead. Such agents must therefore jointly handle language understanding, argument grounding, tool selection, and execution feedback.

Recent benchmarks have substantially advanced the evaluation of function calling, multilingual tool use, and interactive mobile agents. However, these capabilities remain only partially integrated: multilingual function-calling benchmarks primarily assess language-to-tool mapping, while mobile-agent benchmarks increasingly support interactive execution but provide limited coverage of localized multilingual API grounding. In particular, whether lightweight on-device agents can transform *localized multilingual intent* into executable mobile API arguments and reliably recover from dynamic execution feedback remains insufficiently characterized. Figure 1 summarizes this benchmark landscape and positions PolyAgent at the intersection of multilingual intent grounding, API-driven execution, and interactive mobile environments.

To address this gap, we introduce **PolyAgent**, a multilingual diagnostic benchmark for API-driven on-device mobile control. PolyAgent contains over 5,400 expert-validated single-turn queries spanning 6 typologically diverse languages and 47 real-world mobile operating-system APIs. Rather than directly translating an English source corpus, we employ a culturally grounded synthesis and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

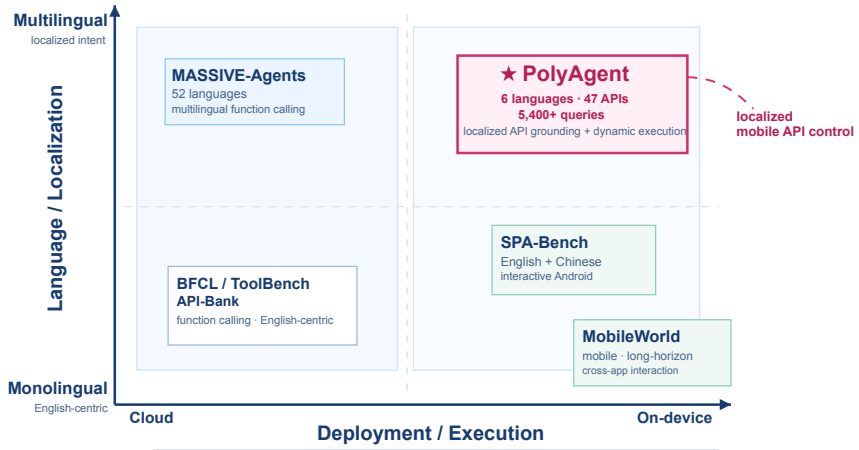


Figure 1: Positioning of PolyAgent in the benchmark landscape. Existing benchmarks address different subsets of multilingual tool use, stateful execution, and mobile-agent interaction, while PolyAgent targets their intersection through localized multilingual API grounding and dynamic mobile execution.

native-expert validation pipeline to evaluate localized intent and argument grounding. Integrated with ToolSandbox, PolyAgent supports interactive execution and evaluates both trajectory-level behavior and environment-state outcomes through Trajectory Precision (TP) and Goal Success Rate (GSR).

Our experiments with lightweight open-source models reveal substantial cross-lingual differences under interactive execution. In particular, we observe that the magnitude of the multilingual performance gap(FitzGerald et al., 2023; Hu et al., 2020) depends on both model family and execution setting: Qwen models exhibit pronounced degradation for several non-English languages under dynamic feedback, whereas Gemma models maintain comparatively stable performance across languages while achieving strong overall accuracy. These results suggest that multilingual robustness in mobile tool use cannot be inferred reliably from English-only or static function-calling evaluation.

In summary, our contributions are three-fold:

1. **Multilingual Mobile API Benchmark**: We introduce PolyAgent, a diagnostic benchmark for localized API-driven mobile control spanning 6 typologically diverse languages, 5,400+ queries, and 47 real-world mobile APIs.
2. **State-Aware Evaluation Protocol**: We integrate PolyAgent with ToolSandbox and evaluate both trajectory-level action fidelity and environment-state outcomes through TP and GSR.
3. **Cross-Lingual Diagnostic Findings**: We systematically evaluate lightweight open-source models under interactive execution and identify substantial model-family- and language-dependent variation in multilingual tool-use robustness.

2 Related Work

Research on agent evaluation has recently progressed along several complementary directions, including function calling, multilingual tool use, stateful tool execution, and mobile-agent interaction. Rather than being isolated capabilities, these directions increasingly overlap; however, their integration for localized, API-driven on-device control remains limited.

Function Calling and Stateful Tool Use. Early function-calling benchmarks such as API-Bank (Li et al., 2023), ToolBench (Qin et al., 2024), and BFCL (Patil et al., 2024a) established standardized evaluation of tool selection and structured argument generation. More recent work has moved beyond isolated call correctness toward interactive execution. ToolSandbox (Lu et al., 2025) introduces stateful tool execution, implicit dependencies between tools, conversational interaction, and dynamic evaluation of intermediate and final milestones. These advances demonstrate that tool-use evaluation benefits from modeling execution state and feedback rather than treating each function

call independently. Nevertheless, these benchmarks primarily evaluate general-purpose tool use rather than localized multilingual control of mobile operating-system APIs.

Multilingual Function Calling. Multilingual evaluation has also expanded substantially. MASSIVE-Agents (Kulkarni et al., 2025) extends function-calling evaluation to 52 languages, 55 functions, and 286 arguments, revealing substantial performance variation across languages. This line of work establishes that language-dependent differences in tool selection and argument generation can persist even when the underlying function schema is shared. However, multilingual function-calling benchmarks generally evaluate whether a model produces the correct structured call, without requiring that the call be executed in a persistent mobile environment where localized entities and parameters can affect subsequent system state.

Mobile-Agent Evaluation. In parallel, mobile-agent benchmarks have developed increasingly realistic interactive environments. AndroidWorld (**?**) provides reproducible Android tasks with programmatic initialization and evaluation, while Mobile-Bench (Deng et al., 2024) evaluates mobile agents across both UI interaction and API-oriented actions. SPA-Bench (Chen et al., 2025) further provides an interactive Android evaluation framework covering system and third-party applications in both English and Chinese, together with task-completion and resource-consumption metrics. More recent benchmarks such as MobileWorld (Kong et al., 2026) extend mobile evaluation to long-horizon, cross-application workflows, agent-user interaction, and hybrid tool use. These efforts substantially broaden the realism of mobile-agent evaluation, but their primary emphasis remains general mobile task completion rather than multilingual intent-to-API grounding.

Toward Localized, API-Driven Mobile Control. The above directions collectively establish important components of modern agent evaluation: structured function calling, multilingual tool use, stateful execution, and interactive mobile environments. Yet they remain only partially integrated. In particular, existing benchmarks provide limited characterization of the complete pipeline

localized multilingual intent $\rightarrow$ *API argument grounding* $\rightarrow$ *mobile execution* $\rightarrow$ *environment-state transition*.

This setting introduces challenges that are not captured by function-call accuracy alone, including culturally localized entities and parameters, schema ambiguity, and recovery from execution feedback. PolyAgent targets this intersection by combining culturally grounded multilingual data, executable mobile APIs, interactive state transitions, and outcome-based evaluation within a unified diagnostic framework.

3 PolyAgent Benchmark Construction and Sandbox Integration

To rigorously evaluate on-device multilingual agents in dynamic operational environments, we construct **PolyAgent**. Unlike static dataset designs, PolyAgent couples culturally localized user intents across 6 typologically diverse languages with executable mobile system APIs integrated into **ToolSandbox**. Fig. 2 illustrates our end-to-end benchmark construction pipeline.

3.1 API Registry and Domain Taxonomy

We establish a standardized registry $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ comprising $M = 47$ real-world mobile operating system APIs across 7 core functional domains: *Clock & Calendar*, *System & Device Settings*, *Communication & Contacts*, *Weather & Environment*, *Camera & Photography*, *Navigation & Travel* and *App Management*. **As illustrated in Fig. 3**, the inner ring specifies these core functional domains, while the outer ring details the functional clusters of the integrated APIs.

To ensure rigorous function calling and **executable API argument binding** evaluation, each API schema $s_i \in \mathcal{S}$ is formally structured as a 4-tuple:

$$s_i = \langle \text{Name}_i, \mathcal{P}_i, \text{RetType}_i, \text{Desc}_i \rangle \tag{1}$$

where Name_i denotes the unique function identifier, RetType_i specifies the return payload type (e.g., `Boolean`, `JSON`), Desc_i provides natural language functional semantics, and $\mathcal{P}_i =$

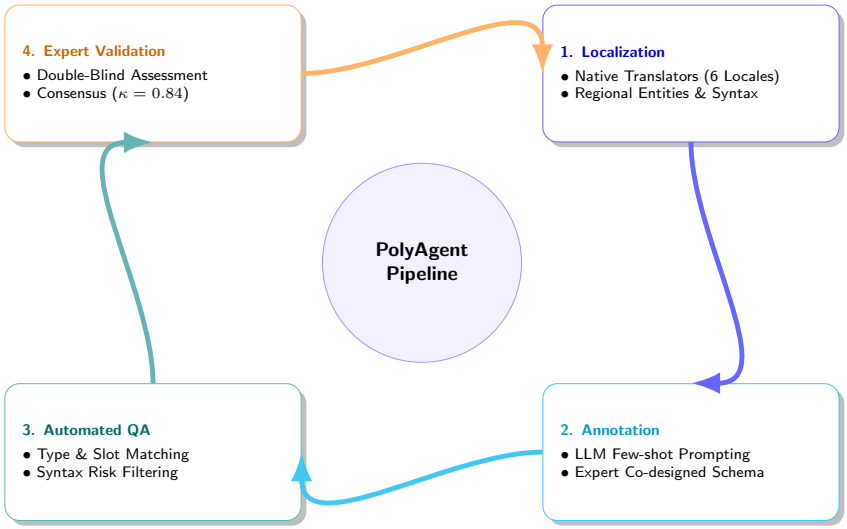


Figure 2: Cyclic Human-in-the-Loop Synthesis Pipeline for PolyAgent.



Figure 3: **Domain taxonomy of API registry $\mathcal{S}$.** Inner ring: 7 core mobile OS domains; outer ring: clusters of $M = 47$ real-world APIs.

$\{(k_j, t_j, r_j, d_j)\}_{j=1}^{|\mathcal{P}_i|}$ represents the parameter specification set. Here, k_j, t_j, r_j, and d_j represent parameter key, **programmatic data type** ($t_j \in \{\texttt{String}, \texttt{Integer}, \texttt{Boolean}, \texttt{Enum}\}$), requirement flag ($r_j \in \{\text{Required}, \text{Optional}\}$), and parameter description, respectively.

The complete registry detailing all 47 OS API schemas is provided in Section A.

3.2 Culturally Localized Synthesis Pipeline

PolyAgent comprises over 5,400 high-quality user queries across 6 typologically diverse languages: Arabic (AR), Thai (TH), Spanish-Latin (ES-LA), Indonesian (ID), Tagalog (TL), and Malay (MS). Because automated translation tools inherently produce ”translationese” and lack cultural OS usage awareness, **human expertise serves as the foundational anchor of our dataset**. We institute a

rigorous four-stage, human-in-the-loop pipeline where native translators and domain experts actively drive data authenticity:

1) Translator-Led Cultural Localization. Spearheaded by native professional translators rather than machine translation, this stage embeds authentic linguistic instincts and regional OS habits. Translators manually adapt seed queries to incorporate regional entities (e.g., local ride-hailing/banking apps), domestic naming conventions, and natural syntax unreachable by automated tools.

2) Expert-Audited Schema Annotation. To map localized queries to executable APIs, candidate function calls are generated via LLM few-shot prompting. Crucially, domain experts co-design all exemplars and audit boundary conditions, ensuring correct multi-argument binding without compromising the nuanced intent crafted by translators.

3) Automated QA for Expert Relief. A multi-layered automated filter handles preliminary syntax, parameter type, and required argument checks. This stage specifically eliminates execution risks upfront to relieve cognitive burden, allowing human experts to focus their energy entirely on deep pragmatic validity.

4) Rigorous Expert Validation & Consensus. As the ultimate guarantor of PolyAgent's gold-standard fidelity, certified native specialists evaluate candidates under a double-blind setup across three dimensions: *Linguistic Authenticity*, *Parameter Solvability*, and *Intent Ambiguity*. Multi-expert consensus resolves edge cases, achieving high inter-annotator agreement (Fleiss' Kappa $\kappa = 0.84$).

3.3 Environment State and ToolSandbox Coupling

To transcend static path matching and enable dynamic closed-loop evaluation, each query X in PolyAgent is coupled with an executable environment configuration in **ToolSandbox**:

- **Initial Environment State (S_0):** Mock mobile databases (e.g., pre-populated contacts, settings, calendar events) instantiated to provide deterministic pre-conditions.
- **Canonical Target Action (a^*):** Defines the ground-truth API function f^* and its strongly-typed argument binding P^*, denoted as $a^* = (f^*, P^*)$.
- **Target Goal State (S^*):** Formally defined system state deltas ($\Delta\text{State} = S^* - S_0$). Executing a^* triggers state transition ($S_0 \xrightarrow{a^*} S^*$), enabling deterministic state verification during closed-loop execution.

3.4 Dataset Statistics and Typological Diversity

Table 1 provides a detailed breakdown of PolyAgent. The benchmark exhibits strong balance across functional domains while reflecting genuine linguistic variations in utterance length and argument density across language families.

Table 1: Overall dataset statistics of PolyAgent across target languages.

Metric	AR	TH	ID	MS	ES-LA	TL
Total Queries	908	879	928	879	948	916
Avg. Arguments per Query	1.33	1.37	1.30	1.30	1.30	1.26
Covered APIs	47	47	47	47	47	47
Avg. Utterance Length (Chars)	38.00	41.49	47.12	49.53	50.79	51.01

4 Task Formulation and Evaluation Metrics

4.1 Task Formulation

Formally, let X denote a natural language user query formulated in a target language $L \in \mathcal{L}$, and $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ denote the registry of available mobile system API schemas. The on-device environment is instantiated as a dynamic sandbox $\mathcal{E}$ with an initial environment state S_0.

Given X and S_0, the agent interacts with $\mathcal{E}$ within an interactive closed loop over up to T execution steps to generate an execution trajectory:

$$\tau = ((a_1, P_1, o_1), (a_2, P_2, o_2), \ldots, (a_T, P_T, o_T)) \tag{2}$$

where a_t represents the invoked API function name at step t, $P_t = \{(k_j, v_j)\}$ represents the parameter dictionary with strongly-typed argument bindings, and $o_t = \mathcal{E}(a_t, P_t)$ is the execution feedback (e.g., return payloads or runtime exceptions) from the sandbox. **Note that $T > 1$ represents interactive runtime error recovery loops driven by sandbox observations o_t under a single-turn user intent X.** The trajectory τ drives the sandbox state transition $S_0 \xrightarrow{\tau} S_T$. The objective is to reach a target goal state S^* that satisfies user intent X.

Pre-conditioned State Guarantee. To isolate the agent's core capabilities in intent selection and structured API argument binding, we operate under a closed-world state setup. Formally, for every ground-truth parameter value v_j^* required to fulfill query X, the initial sandbox state S_0 is guaranteed to satisfy $v_j^* \in \text{Domain}(S_0)$ (e.g., target contacts, system configuration flags, or media files referenced in X are pre-populated in S_0). Consequently, any execution failure in $\mathcal{E}$ strictly reflects deficits in agent planning, API mapping, or type-constrained parameter extraction, rather than missing entity pre-conditions in the environment.

4.2 Closed-Loop Evaluation Framework

To account for dynamic environment feedback and execution-level error recovery, we evaluate agents along two complementary dimensions:

(1) Trajectory Precision (TP) assesses whether the step-wise actions adhere to the canonical target invocation $a^* = (f^*, P^*)$:

$$\text{TP} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left((a_1, P_1)_i \equiv (f^*, P^*)_i\right) \tag{3}$$

capturing whether the agent correctly selects the target API and binds all arguments on its initial execution attempt without extraneous or incorrect calls.

(2) Goal Success Rate (GSR) evaluates whether the final environment state S_T satisfies the target goal state S^*, measuring state change deltas $\Delta\text{State} = S_T - S_0$. GSR combines deterministic programmatic checks with semantic verification:

$$\text{GSR} = \frac{1}{N} \sum_{i=1}^{N} \Psi\left(S_{T,i}, S_i^*; X_i\right) \tag{4}$$

where $\Psi(\cdot) \in \{0, 1\}$ denotes binary goal state fulfillment, verified via programmatic database deltas for deterministic APIs and an evaluator LLM ($\mathcal{M}_{\text{judge}}$) for soft semantic outcomes.

4.3 LLM-as-a-Judge Protocol in ToolSandbox

Even within an execution sandbox, an evaluator LLM ($\mathcal{M}_{\text{judge}}$) is indispensable for validating non-deterministic or semantically flexible system outcomes. We invoke $\mathcal{M}_{\text{judge}}$ under two specific scenarios: **(1) Soft-State Verification**, when target APIs modify free-form text (e.g., reminder titles, event notes) where exact string equality $S_T = S^*$ is overly restrictive; and **(2) Flexible Argument Alignment**, when API execution involves fuzzy matching (e.g., localized contact search queries) requiring natural language semantic interpretation.

All judge evaluations execute under strict zero-shot prompting with zero temperature ($T = 0$) to guarantee deterministic and reproducible evaluation.

5 Experiments and Empirical Diagnostics

In this section, we present the empirical evaluation of edge-deployed LLM agents within **ToolSandbox** across seven diverse languages. Our analysis is structured around two core research questions:

Table 2: **Main experimental results across 6 localized target languages in ToolSandbox. TP** (%) denotes initial Trajectory Precision ($t = 1$), and **GSR** (%) measures final Goal Success Rate under closed-loop execution budget ($T \leq 3$). The English baseline (EN) serves as the capability anchor. **Target Avg.** is strictly computed across the 6 localized target languages. Bold indicates top performance within each model tier.

Model	Anchor Baseline		Localized Target Languages ($\mathcal{L}_{\text{target}}$)												Target Avg.	
	EN		AR		TH		ID		MS		ES-LA		TL		TP	GSR
	TP	GSR	TP	GSR	TP	GSR	TP	GSR	TP	GSR	TP	GSR	TP	GSR	TP	GSR
Larger-Scale Controls (Validation Baseline)																
Qwen-3.6-27B	78.51	92.76	64.74	84.25	54.22	79.18	71.36	90.30	67.21	84.85	72.43	88.40	62.57	79.69	65.42	84.45
Gemma-4-31B-it	**86.56**	**95.20**	**84.17**	**92.51**	**81.96**	**94.65**	**85.89**	**94.50**	**83.04**	**91.95**	**83.68**	**94.51**	**82.40**	**92.79**	**83.52**	**93.49**
Edge-Deployed Models (Primary Target)																
Qwen-3.5-2B	46.09	68.90	36.15	42.51	15.15	26.51	38.30	52.37	37.83	47.65	41.62	55.91	26.47	39.19	32.59	44.02
Qwen-3.5-4B	**66.60**	**82.60**	57.87	68.17	31.87	46.19	58.13	77.05	54.96	71.16	59.30	75.53	42.38	58.52	50.75	66.10
Gemma-4-E4B-it	65.07	79.92	**65.51**	**77.20**	**60.20**	**77.70**	**70.09**	**85.67**	**66.68**	**82.34**	**63.19**	**79.85**	**59.11**	**73.14**	**64.13**	**79.32**

- **RQ1 (Cross-Lingual Transfer):** How effectively do lightweight, edge-deployed LLM agents transfer tool-use capabilities from English to non-Latin and low-resource languages?
- **RQ2 (Closed-Loop Dynamics & Evaluation Gap):** Why does dynamic closed-loop goal success ($\text{GSR}_{T\leq 3}$) systematically exceed static single-step precision ($\text{TP}_{t=1}$), and what roles do error recovery and functional equivalence play?

5.1 Experimental Setup

Evaluated Models. We evaluate five open-source models across two scales. The primary edge-deployed models are **Qwen-3.5-2B**(Qwen Team, 2026a), **Qwen-3.5-4B**, and **Gemma-4-E4B-it**(Team, 2026), while **Qwen-3.6-27B**(Qwen Team, 2026b) and **Gemma-4-31B-it** serve as larger-scale controls. All models are evaluated in a zero-shot function-calling setting with standardized system prompts.

Sandbox and Evaluation. All evaluations are conducted in **ToolSandbox** from the pre-conditioned state S_0, with a maximum of $T_{\max} = 3$ execution steps per query. `DeepSeek-V4-Flash`(DeepSeek-AI, 2026) is used as the zero-shot LLM judge with temperature $T = 0$ for soft-state verification and argument alignment.

Languages. We evaluate six localized target languages, $\mathcal{L}_{\text{target}} = \{\text{AR, TH, ES-LA, ID, TL, MS}\}$, together with English (En_{anchor}) as the reference anchor. All target-language averages are computed exclusively over the six localized languages.

5.2 Main Quantitative Results

Table 2 presents the primary performance metrics across English (En_{anchor}) and six localized target languages. We observe three primary macro trends:

Key empirical observations from Table 2 include:

- **Cross-Lingual Degradation (RQ1):** Performance degrades noticeably from English to non-Latin target scripts. For example, Qwen-3.5-2B's TP drops from 46.09% in English to 15.15% in Thai. However, **Gemma-4-E4B-it** shows remarkable stability, maintaining 60.20% TP in Thai and outperforming Qwen-3.5-4B by over 28 percentage points.
- **The Closed-Loop Advantage (RQ2):** Across all models, final GSR systematically exceeds initial TP (by **14.19%** on average for non-English targets). This disparity highlights two distinct advantages of dynamic sandbox evaluation: (1) agents effectively leverage real-time execution feedback for multi-turn error recovery, and (2) dynamic verification (GSR) correctly credits functionally equivalent tool parameters that rigid static matching (TP) incorrectly penalizes.

5.3 Cross-Lingual Gap: Qwen vs. Gemma under Dynamic Feedback

A key empirical finding is the contrasting cross-lingual resilience between Qwen and Gemma when processing interactive feedback:

Multilingual Degradation of Qwen under Dynamic Feedback. While Qwen-3.5-4B performs well in English (82.60% GSR / 66.60% TP), its performance drops significantly when handling non-English dynamic queries—suffering a 36.41% drop in Thai (46.19% GSR) and 24.08% in Tagalog (58.52% GSR). Qualitative inspection shows that under non-Latin dynamic feedback in ToolSandbox, Qwen-3.5-4B suffers from severe schema boundary degradation (TP $= 31.87\%$ in TH). It frequently fails to parse environment error logs, producing hallucinated argument formats or translated enum keys that disrupt multi-turn self-correction.

Cross-Lingual Stability and Closed-Loop Execution in Gemma. Conversely, the Gemma family demonstrates strong stability across both trajectory precision (TP) and goal fulfillment (GSR). Gemma-4-E4B-it achieves a target language average GSR of 79.32% and TP of 64.13%, outperforming Qwen-3.5-4B by $+13.22\%$ and $+13.38\%$ respectively. Remarkably, its cross-lingual capability remains highly resilient: the GSR gap between English (79.92%) and Thai (77.70%) is only 2.22%. The result suggests that model architecture and training may influence schema adherence and feedback handling.

5.4 Single-Step Precision vs. Dynamic Sandbox Execution (RQ2)

To answer **RQ2**, we quantify the gap between static single-step trajectory precision ($\text{TP}_{t=1}$) and dynamic closed-loop goal success ($\text{GSR}_{T\leq 3}$) directly using Table 2.

Across all target non-English languages, dynamic closed-loop evaluation (GSR) consistently exceeds static single-step precision (TP) by an average of **14.19%** (and up to **19.03%** on Qwen-3.6-27B). Qualitative trajectory inspection confirms three core drivers for this divergence:

1. **Closed-Loop Error Self-Correction:** Static evaluation (TP) instantly penalizes initial syntax errors or wrong parameter bindings at $t = 1$. In contrast, **ToolSandbox** provides real-time environment error logs that allow agents to recover and fulfill the task in subsequent turns.
2. **Natural Language Parameter Resilience:** Open-ended parameters (e.g., `message_body`, `search_query`) naturally admit valid semantic paraphrasing that fails rigid initial checking but executes seamlessly in the sandbox.
3. **Functional Trajectory Equivalence:** Agents frequently discover alternative, unannotated API pathways that achieve identical target state changes (ΔState), which single-step matching misclassifies as failures.

This gap highlights that static single-step evaluation is often overly conservative, validating the necessity of state-level interactive verification in multilingual agent benchmarks.

5.5 Error Diagnostics and Failure Taxonomy

To understand how tool execution breaks down across model architectures, we analyze the relative distribution of failure modes conditioned on all recorded failure trajectories ($N = 8,314$) across both English (En_{anchor}) and six target non-English languages. We categorize errors into three primary modes: *Dynamic Observation Drift* (DOD, failing to invoke tools under ambiguous or error contexts), *API Selection Misalignment* (API, choosing semantically adjacent tools), and *Parameter Error* (Param, argument string mismatches due to natural language parameter paraphrasing in English or over-translation in target languages). Syntax parse errors ($n = 6$) and hallucinations ($n = 3$) remain negligible ($< 0.1\%$).

Key Diagnostic Takeaways. As shown in Table 3, analyzing conditional error behavior yields three core insights:

1. **Conservative Behavior in Lightweight Gemma: Gemma-4-E4B-it** exhibits a distinct failure profile, with **41.80%** of its errors originating from Dynamic Observation Drift (DOD)—more than double the rate of equivalent Qwen SLMs (16–22%). Under interactive execution feedback, Gemma-E4B prefers falling back to natural language text rather than executing uncertain API calls. As Gemma scales to 31B, DOD drops to 22.89%, aligning with broader industry baselines.
2. **Parameter Mismatch as a Dominant Failure Mode:** Across nearly all models (Qwen variants and Gemma-31B), **Parameter Error accounts for over 50% of total failures** (averaging 50.29% overall). While in English this stems from open-ended natural language paraphrasing, in

Table 3: Model-wise failure mode taxonomy conditioned on failed trajectories ($N = 8,314$) across all evaluated languages (including En_{anchor}). Percentages reflect $P(\text{Error Mode} \mid \text{Failure})$. Note that primary category sums ($n = 8,307$) exclude negligible syntax/hallucination failures ($n = 7$).

Model	Total Failed	DOD (No-Call)	API Selection	Parameter Error
Qwen-3.5-2B	3,480	576 (16.55%)	1,011 (29.05%)	1,891 (54.34%)
Qwen-3.5-4B	2,080	470 (22.60%)	537 (25.82%)	1,072 (51.54%)
Gemma-4-E4B-it	1,390	581 (**41.80%**)	321 (23.09%)	487 (35.04%)
Qwen-3.6-27B	949	184 (19.39%)	240 (25.29%)	523 (**55.11%**)
Gemma-4-31B-it	415	95 (22.89%)	111 (26.75%)	208 (50.12%)
Overall	**8,314**	**1,906 (22.92%)**	**2,220 (26.70%)**	**4,181 (50.29%)**

non-English scripts it manifests as severe verbatim string translation leakage (e.g., erroneously translating fixed system Enum keys or contact names into target scripts).

3. **Invariance of API Selection Precision:** Across all five architectures, API Selection Misalignment remains tightly bounded between **23.09% and 29.05%**. Scaling model size significantly reduces absolute failure counts (e.g., 3,480 in Qwen-2B vs. 949 in Qwen-27B), but the relative proportion of tool selection errors remains Invariance of API Selection Precision.

6 Conclusion and Future Work

In this work, we presented **PolyAgent**, a comprehensive multilingual benchmark for API-driven mobile device control evaluated through closed-loop, dynamic execution environments. By moving beyond rigid static trajectory matching, **PolyAgent** provides a realistic evaluation framework that assesses state-level goal reachability, cross-lingual robustness, and resilience to interactive runtime observations across diverse edge-deployed SLMs.

Our empirical diagnostics across 8,314 execution breakdowns yield three core insights for the agent community:

1. **Imperative for Dynamic Closed-Loop Evaluation:** Static single-step evaluation ($\text{TP}_{t=1}$) systematically underestimates true agent task fulfillment ($\text{GSR}_{T\leq 3}$) by an average of **14.19%** (and up to **19.03%**), blinded by functional trajectory equivalence and valid parameter paraphrasing. Closed-loop state verification in **PolyAgent** is indispensable for realistic mobile agent assessment.
2. **Parameter Localization as the Cross-Lingual Bottleneck:** Performance drops significantly in non-Latin scripts (e.g., a **36.41%** GSR degradation in Thai for Qwen-3.5-4B). Crucially, over **50%** of total failures (50.29%) stem from parameter-level over-translation (verbatim string leakage), whereas syntax parse errors and hallucinations are virtually negligible ($< 0.1\%$).
3. **Architectural Trade-offs in Interactive Recovery:** Dynamic Observation Drift (DOD) accounts for **22.92%** of total failures, exhibiting strong architectural variation—lightweight models like Gemma-4-E4B-it display a conservative fallback bias (41.80% DOD), preferring natural language text over executing uncertain tools.

Future Work. We plan to extend **PolyAgent** in two primary directions: (1) integrating *procedural schema perturbation and dynamic contamination defenses* to ensure long-term benchmark longevity, and (2) expanding the sandbox topology to support *multi-agent collaborative mobile device environments* for evaluating complex cross-lingual tool coordination.

REFERENCES

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Yixing Li, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye Hao, Jun Wang, and Kun Shao. Spa-bench: a comprehensive benchmark for smartphone agent evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=OZbFRNhpwr.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, and Shuo Shang. Mobile-bench: An evaluation benchmark for llm-based mobile agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 8813–8831. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.478. URL https://doi.org/10.18653/v1/2024.acl-long.478.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gökhan Tür, and Prem Natarajan. MASSIVE: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 4277–4302. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.235. URL https://doi.org/10.18653/v1/2023.acl-long.235.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080, 2020. URL https://arxiv.org/abs/2003.11080.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. Mobile-world: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 6142–6167. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.278. URL https://doi.org/10.18653/v1/2026.acl-long.278.

Mayank Kulkarni, Vittorio Mazzia, Judith Gaspers, Chris Hench, and Jack FitzGerald. Massive-agents: A benchmark for multilingual function-calling in 52 languages. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 20193–20215. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-EMNLP.1099. URL https://doi.org/10.18653/v1/2025.findings-emnlp.1099.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.187. URL https://doi.org/10.18653/v1/2023.emnlp-main.187.

Zechun Liu, Changsheng Zhao, Forrest N. Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria

Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32431–32454. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/liu24ce.html.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-NAACL.65. URL https://doi.org/10.18653/v1/2025.findings-naacl.65.

Microsoft. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024. doi: 10.48550/ARXIV.2404.14219. URL https://doi.org/10.48550/arXiv.2404.14219.

Shishir G. Patil, Tianjun Zhang, Marcos Z. Sheng, Joseph E. Gonzalez, and Ion Stoica. Berkeley function calling leaderboard. https://gorilla.cs.berkeley.edu/leaderboard.html, 2024a. UC Berkeley Gorilla Team.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *CoRR*, abs/2401.16158, 2024. doi: 10.48550/ARXIV.2401.16158. URL https://doi.org/10.48550/arXiv.2401.16158.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International*

Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *CoRR*, abs/2312.13771, 2023. doi: 10.48550/ARXIV.2312.13771. URL https://doi.org/10.48550/arXiv.2312.13771.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *CoRR*, abs/2401.02385, 2024. doi: 10.48550/ARXIV.2401.02385. URL https://doi.org/10.48550/arXiv.2401.02385.

A APPENDIX

Due to space constraints in the main manuscript, **??** in this appendix provides the concise catalog of all 47 APIs, while Section A.1 illustrates representative complete schema specifications. The unabridged JSON specifications and extended descriptions for all 47 APIs will be fully anonymized and released in our open-source repository upon publication.

A.1 REPRESENTATIVE COMPLETE SCHEMA SPECIFICATIONS

To illustrate the structural spectrum and schema constraints in PolyAgent, we present complete, unabridged JSON specifications for four representative API categories across different complexity levels below.

1) Simple Single-Argument API: `UninstallApp` (App Management).

```
{
  "name": "UninstallApp",
  "description": "Used to uninstall installed applications from a device. This tool only
     supports the type of application and its generalizations (e.g., application, software,
     program, APP, quick app, mini program, etc.). Users can uninstall a category of
     applications or specific applications by including generalized terms for the application
     type, or by specifying an application name to uninstall a specific application. If
     generalized terms for the above type are not explicitly included and there is no
     specific application name that can be accurately determined, this tool is not
     applicable",
  "parameters": {
    "type": "object",
    "properties": {
      "appName": {
        "type": "string",
        "description": "Extract the exact application name or category description verbatim
     from the user's input. Do not translate, normalize, or infer canonical names. Preserve
     the original language, casing, and all functional descriptors or version suffixes
     present in the query."
      },
      "type": {
        "type": "string",
        "description": "Only includes the following enumerated values: application
     (application type, e.g., application, software, program, APP, quick app, mini program,
     etc.), game (game type, e.g., game, mobile game, online game, web game, PC game, etc.),
     metaService (meta-service type), fa (FA type, e.g., FA, FA service, etc.). Only output
     this parameter if the user's input explicitly contains a direct mention or translation
     of these type terms. Do NOT infer, auto-fill, or assume based on context or app name.",
        "enum": [
          "application",
          "game",
          "metaService",
          "fa"
        ]
      }
    },
    "required": ["appName"]
  }
}
```

2) Complex Multi-Argument API (Calendar): `CreateSchedule` (Clock Calendar).

```
{
  "name": "CreateSchedule",
  "description": "Creating a schedule refers to adding, incorporating information about
     things to do, activities, or meetings into a schedule or calendar, or remembering,
     adding, incorporating schedules from the screen or phone. It allows setting the title
     and content of the schedule, the time of the schedule; setting the frequency and
     scenario for schedule reminders; setting keywords and indexes for extracting schedules",
  "parameters": {
    "type": "object",
    "properties": {
      "event": {
        "type": "string",
        "description": "The core activity for the schedule. Extract the exact phrase from the
     query in its original language. Preserve essential distinguishing modifiers (e.g.,
     'Christmas dinner'), but strip redundant generic locations/times if they overlap with
     other fields. Examples: waking up, going out, eating, etc. Do not translate or
     normalize."
      },
      "frequency": {
        "type": "string",
```

```
        "description": "The recurrence or reminder frequency for the schedule. Includes the
      following enumeration values: every (every time, each time), once (next time, this time,
      next time). Only annotate this slot when the query contains explicit words or
      descriptions indicating the recurrence or frequency; otherwise, leave it unannotated and
      do not assume once by default.",
        "enum": [
          "every",
          "once"
        ]
      },
      "isPicture": {
        "type": "string",
        "description": "Set to true if the user explicitly mentions visual sources like
      'screen, picture, photo, interface' and indicates the schedule should be extracted from
      them; otherwise, do not output",
        "enum": [
          "true"
        ]
      },
      "objectReference": {
        "type": "string",
        "description": "Set to true if the user uses pronouns (e.g., 'this', 'that') to refer
      to a schedule source, and historical context confirms it points to a previously shared
      image, screen capture, or visual file; otherwise, do not output",
        "enum": [
          "true"
        ]
      },
      "scene": {
        "type": "string",
        "description": "The scenario that triggers the schedule reminder, includes the
      following enumerated values: onCar (getting on the car), offCar (getting off the car),
      stopCar (parking the car), toHome (arriving home), toCompany (arriving at the company)",
        "enum": [
          "onCar",
          "offCar",
          "stopCar",
          "toHome",
          "toCompany"
        ]
      },
      "index": {
        "type": "string",
        "description": "The index value specified by the user when extracting a schedule from
      a picture, for example: first, 1st, second, etc"
      },
      "time": {
        "type": "string",
        "description": "The time for the schedule. Extract the exact time expression as
      stated in the query, preserving the original language and phrasing. Avoid vague ranges;
      use clear time descriptions like specific points, dates, or relative times (e.g., '6
      o'clock, 14:30, morning, the 3rd, yesterday, Monday, weekend, last week, next Tuesday,
      this weekend, early part of the month, beginning of the month, April, last month, the
      3rd quarter, spring, 2024, 2023-11-15, last year, 3 days later, 3 hours ago, National
      Day, the first day of the first lunar month, Grain Rain'). It can also be relative to a
      point, such as 'before 8 o'clock, after 9 o'clock'. If numeric formats like '7.30'
      appear, standardize to '7:30', but otherwise keep the original query's wording intact."
      },
      "timeRange": {
        "type": "string",
        "description": "The time range for the schedule specified by the user, includes the
      following enumerated value: wholeDay (all day, the whole day, entire day)",
        "enum": [
          "wholeDay"
        ]
      },
      "referenceContent": {
        "type": "string",
        "description": "Reference content, output format: ${id.xxx}, where xxx represents the
      id value of the picture, file, or text sent by the user in the historical conversation.
      Only the above three types of original user inputs are allowed to be referenced; do not
      reference tool execution results from previous turns"
      }
    },
    "required": ["event", "time"]
  }
}
```

3) Complex Multi-Argument API (Communication): `SendMessage` (Communication Contacts).

```
{
  "name": "SendMessage",
  "description": "Sending text messages to others via SMS or social applications. Common
     social applications include SMS, WeChat, QQ, Line, Messenger, etc. Users need to clearly
     define their intention for sending the message and can specify the recipient and content
     of the message.",
  "parameters": {
    "type": "object",
    "properties": {
      "appName": {
        "type": "string",
        "description": "Application used for sending messages, common examples include SMS,
     WeChat, QQ, Line, Messenger, etc. Extract ONLY when explicitly stated as a specific
     channel/proper noun. Do NOT extract when used as a verb or generic noun (e.g., 'SMS my
     mom', 'send an SMS'). Preserve exact original script, case, and punctuation."
      },
      "contact": {
        "type": "string",
        "description": "Message recipient, which can be an individual, group, channel, or
     official account in different applications. Extract the phrase exactly as written,
     including all determiners, possessives, and titles. Preserve original script, case, and
     punctuation. Do not infer or compress."
      },
      "content": {
        "type": "string",
        "description": "Core textual payload to be conveyed. Extract ONLY the direct message
     intent: strip meta-instructions and greetings, and where appropriate, convert indirect
     requests into direct second-person address. If not explicitly stated, leave empty. Do
     NOT generate placeholder text or conversational expansions. Preserve original script,
     case, and punctuation."
      },
      "phoneNumber": {
        "type": "string",
        "description": "Message recipient identified by phone number. Extract exactly as
     stated, preserving original format. Leave empty if not provided."
      },
      "objectReference": {
        "type": "string",
        "description": "Whether it is a referential expression, such as 'this', 'that', 'he',
     'she', 'it', etc. Output 'true' if present.  If not mentioned, omit this field.",
        "enum": [
          "true"
        ]
      }
    },
    "required": []
  }
}
```

4) Easily Confusable / State-Bound API: `ModifySchedule` (Clock Calendar).

```
{
  "name": "ModifySchedule",
  "description": "Modifies an existing schedule or calendar event (e.g., rescheduling,
     postponing, or updating event details). Supports searching for target schedules by time
     or title before applying updates. Note: If a schedule tool was used to create or query
     an entry in previous contexteven if referred to as a \"reminder\"use this tool to modify
     it.",
  "parameters": {
    "type": "object",
    "properties": {
      "sourceTime": {
        "type": "string",
        "description": "The original time of the schedule to be modified, such as expressions
     like '6 o'clock', 'tomorrow', '7 am', etc. Do not translate to English or force ISO/24h
     formats. If there are expressions like '7.30' or '9.30' that are clearly time references
     in context, output them in standard time format like '7:30' or '9:30'."
      },
      "sourceEvent": {
        "type": "string",
        "description": "The event condition used to search for the schedule, such as
     expressions like 'meeting', 'doing homework', etc. Extract only the core event noun or
     verb phrase. Retain location, purpose, or other modifiers only if they are integral to
     the schedule's official name. Preserve the original language."
      },
      "targetTime": {
        "type": "string",
```

```
        "description": "The modified schedule time, which can be a specific time point or
    date, such as expressions like '6 o'clock', 'tomorrow', '7 am', etc. If there are
    expressions like '7.30' or '9.30' that are clearly time references in context, output
    them in standard time format like '7:30' or '9:30'. Only include if explicitly changed;
    omit this field entirely if the time remains unchanged."
      },
      "targetEvent": {
        "type": "string",
        "description": "The modified schedule event, such as expressions like 'meeting',
    'doing homework', etc. Only include if explicitly changed to a different event; omit
    this field entirely if the event remains unchanged. Do not copy the sourceEvent value.
    Extract only the core event phrase. Retain location, purpose, or other modifiers only if
    they are integral to the schedule's official name. Preserve the original language."
      },
      "entityId": {
        "type": "string",
        "description": "The unique identifier of the existing schedule, used to precisely
    locate the target schedule to be modified. This field should only be filled if the
    entityId information of the schedule was returned in previous tool interactions; if no
    entityId was provided in prior steps, this field should be left blank"
      }
    },
    "required": [sourceEvent, "sourceTime"]
  }
}
```

A.2 Full Catalog of 47 Mobile OS APIs

Table 4: Full catalog of all 47 mobile OS APIs in PolyAgent.

Domain	Function Name	Brief Description & Parameters
System & Device Settings	`TurnOffVibrationMode`	Disable device vibration mode. *Args:* `deviceType` (str, opt)
App Management	`BackApp`	Switches back to specified application interface. *Args:* `appName` (str), `type` (str, opt)
Camera & Photography	`CloseCamera`	Terminates camera application and recording session. *Args:* `deviceType` (str, opt), `mode` (str, opt), `roomType` (str, opt), `subDeviceType` (str, opt), `device` (str, opt)
App Management	`UninstallApp`	Uninstalls specific or generalized applications from a device. *Args:* `appName` (str), `type` (str, opt)
Camera & Photography	`TakeSelfie`	Opens camera interface for taking selfies. *Args:* `device` (str, opt), `deviceType` (str, opt), `mode` (str, opt), `roomType` (str, opt), `time` (str, opt)
Clock & Calendar	`ModifySchedule`	Modifies existing schedules or calendar events by time or title. *Args:* `sourceTime` (str), `sourceEvent` (str), `targetTime` (str, opt), `targetEvent` (str, opt), `entityId` (str, opt)
System & Device Settings	`TurnOnNoDisturb`	Activates Do Not Disturb for specific time periods. *Args:* `deviceType` (str, opt), `type` (str, opt), `time` (str, opt), `duration` (str, opt), `targetPage` (str, opt)

Table 4: Full catalog of all 47 mobile OS APIs in PolyAgent.

Domain	Function Name	Brief Description & Parameters
System & Device Settings	`TurnUpVolume`	Adjust device volume via user-specified values and types. *Args:* `deviceType` (str, opt), `space` (str, opt), `subDeviceType` (str, opt), `fraction` (str, opt), `level` (str, opt), `multiple` (str, opt), `number` (str, opt), `percent` (str, opt), `roomType` (str, opt), `volumeType` (str), `volumeDirection` (str, opt), `floor` (str, opt), `destinationSeatPosition` (str, opt)
System & Device Settings	`TurnOnSetting`	Enable specific device setting functions. *Args:* `settingItem` (str), `deviceType` (str, opt), `roomType` (str, opt), `destinationSeatPosition` (str, opt), `rangeType` (str, opt)
Clock & Calendar	`DeleteSchedule`	Removes scheduled events by title, time, or scope. *Args:* `event` (str), `rangeType` (str, opt), `time` (str, opt), `entityId` (str, opt)
Clock & Calendar	`ModifyReminder`	Modifying scheduled reminders by editing time or event details. *Args:* `sourceTime` (str, opt), `sourceEvent` (str), `targetTime` (str), `targetEvent` (str, opt), `scene` (str, opt)
System & Device Settings	`CheckSetting`	Query electronic device setting status. *Args:* `settingItem` (str), `deviceType` (str, opt), `roomType` (str, opt), `destinationSeatPosition` (str, opt)
Clock & Calendar	`CheckSchedule`	Retrieves existing schedule arrangements by title, time, or range. *Args:* `event` (str, opt), `rangeType` (str, opt), `time` (str, opt), `entityId` (str, opt)
Camera & Photography	`OpenCamera`	Launches camera application or specific shooting modes. *Args:* `appName` (str, opt), `device` (str, opt), `deviceType` (str, opt), `mode` (str, opt), `roomType` (str, opt), `subDeviceType` (str, opt), `time` (str, opt)
System & Device Settings	`TurnDownVolume`	Adjusts device volume to user-specified value and type. *Args:* `deviceType` (str, opt), `space` (str, opt), `subDeviceType` (str, opt), `fraction` (str, opt), `level` (str, opt), `multiple` (str, opt), `number` (str, opt), `percent` (str, opt), `roomType` (str, opt), `volumeType` (str), `volumeDirection` (str, opt), `destinationSeatPosition` (str, opt), `floor` (str, opt)
Clock & Calendar	`DeleteReminder`	Delete reminders via title, time, scope, status, and scenario queries. *Args:* `event` (str), `rangeType` (str, opt), `scene` (str, opt), `state` (str, opt), `time` (str, opt)
System & Device Settings	`SetPowerSavingMode`	Opens power-saving mode settings to extend battery life. *Args:* `deviceType` (str, opt), `type` (str, opt)

Table 4: Full catalog of all 47 mobile OS APIs in PolyAgent.

Domain	Function Name	Brief Description & Parameters
Clock & Calendar	`CreateAlarm`	Creates one-time or recurring alarms for task reminders. *Args:* `event` (str, opt), `time` (str), `deviceType` (str, opt)
System & Device Settings	`SetWlan`	Configure wireless local area network functionality. *Args:* `deviceType` (str, opt), `type` (str, opt), `destinationSeatPosition` (str, opt)
System & Device Settings	`TurnOnPowerSavingMode`	Activate power saving mode to extend battery life. *Args:* `deviceType` (str, opt), `type` (str, opt), `mode` (str, opt)
Communication & Contacts	`CheckCallRecord`	View call records including missed calls. *Args:* `deviceType` (str, opt), `roomType` (str, opt)
Clock & Calendar	`CheckReminder`	Query reminders by attributes or open reminder page. *Args:* `event` (str, opt), `rangeType` (str, opt), `scene` (str, opt), `state` (str, opt), `time` (str, opt)
System & Device Settings	`TurnOffWlan`	Disable Wireless Local Area Network functionality. *Args:* `deviceType` (str, opt), `destinationSeatPosition` (str, opt)
Clock & Calendar	`CreateSchedule`	Schedule creation with metadata, reminders, and indexing. *Args:* `event` (str), `frequency` (str, opt), `isPicture` (str, opt), `objectReference` (str, opt), `scene` (str, opt), `index` (str, opt), `time` (str), `timeRange` (str, opt), `referenceContent` (str, opt)
System & Device Settings	`SetVolume`	Set device volume or open settings page. *Args:* `deviceType` (str, opt), `space` (str, opt), `subDeviceType` (str, opt), `degree` (str, opt), `fraction` (str, opt), `level` (str, opt), `multiple` (str, opt), `number` (str, opt), `percent` (str, opt), `roomType` (str, opt), `volumeType` (str), `volumeDirection` (str, opt), `floor` (str, opt), `rangeType` (str, opt), `destinationSeatPosition` (str, opt)
Communication & Contacts	`Redial`	Redials the last dialed number only. *Args:* `deviceType` (str, opt), `roomType` (str, opt)
System & Device Settings	`TurnOnWlan`	Enable Wireless Local Area Network connectivity. *Args:* `deviceType` (str, opt), `type` (str, opt), `destinationSeatPosition` (str, opt)
System & Device Settings	`TurnOffPowerSavingMode`	Disable power saving mode to extend battery life. *Args:* `deviceType` (str, opt), `type` (str, opt), `mode` (str, opt)
Clock & Calendar	`ModifyTimer`	Timer duration modification. *Args:* `duration` (str)
System & Device Settings	`SetSetting`	Configure device settings or open settings page. *Args:* `settingItem` (str), `deviceType` (str, opt), `roomType` (str, opt), `destinationSeatPosition` (str, opt)

Table 4: Full catalog of all 47 mobile OS APIs in PolyAgent.

Domain	Function Name	Brief Description & Parameters
Clock & Calendar	`ResetTimer`	Resets timer to original start without altering duration. *Args:* `deviceType` (str, opt), `roomType` (str, opt)
Communication & Contacts	`SendMessage`	Sending text messages via SMS or social applications. *Args:* `appName` (str, opt), `contact` (str, opt), `content` (str, opt), `phoneNumber` (str, opt), `objectReference` (str, opt)
Clock & Calendar	`OpenAlarm`	Enables existing disabled alarms by identifier, time, or range. *Args:* `event` (str, opt), `rangeType` (str, opt), `time` (str), `index` (str, opt), `entityId` (str, opt), `deviceType` (str, opt)
Communication & Contacts	`Call`	Universal real-time voice and video calling interface. *Args:* `appName` (str, opt), `callMode` (str, opt), `contact` (str, opt), `targetDeviceType` (str, opt), `personalPronoun` (str, opt), `phoneNumber` (str, opt), `phoneCard` (str, opt), `phoneNumberPrefix` (str, opt), `phoneNumberSuffix` (str, opt), `objectReference` (str, opt), `isPicture` (str, opt), `index` (str, opt), `sourceDeviceType` (str, opt), `isSatellite` (str, opt)
Clock & Calendar	`PauseTimer`	Query and control timers by device and room. *Args:* `deviceType` (str, opt), `roomType` (str, opt)
Clock & Calendar	`ResumeTimer`	Resumes paused timers from last state. *Args:* `deviceType` (str, opt), `roomType` (str, opt)
Communication & Contacts	`CheckContact`	Retrieves phone numbers for local contacts and external entities. *Args:* `contact` (str), `deviceType` (str, opt), `roomType` (str, opt), `index` (str, opt)
Weather & Environment	`CheckHumidity`	Retrieves current and forecasted humidity for specific locations and times. *Args:* `destinationLocation` (str, opt), `deviceType` (str, opt), `roomType` (str, opt), `time` (str, opt)
Navigation & Travel	`Navigate`	Triggers navigation app for explicit or ambiguous routing requests. *Args:* `appName` (str, opt), `isReNavigate` (str, opt), `destinationLocation` (str, opt), `departureLocation` (str, opt), `index` (str, opt), `pathPoint` (str, opt), `avoidPoint` (str, opt), `transportationMode` (str, opt), `routeType` (str, opt)
Weather & Environment	`CheckTemperature`	Query current and forecast temperatures for specific areas and times. *Args:* `destinationLocation` (str, opt), `deviceType` (str, opt), `roomType` (str, opt), `time` (str, opt)

Table 4: Full catalog of all 47 mobile OS APIs in PolyAgent.

Domain	Function Name	Brief Description & Parameters
Clock & Calendar	`CancelAlarm`	Disables enabled alarms without removing them from the list. *Args:* `event` (str, opt), `rangeType` (str, opt), `time` (str), `index` (str, opt), `entityId` (str, opt), `deviceType` (str, opt)
Weather & Environment	`CheckWeatherAlert`	Query location-specific weather alerts and severe weather events. *Args:* `destinationLocation` (str, opt), `deviceType` (str, opt), `roomType` (str, opt), `time` (str, opt), `alertType` (str)
Weather & Environment	`CheckWeather`	Location-based weather query and broadcast service. *Args:* `destinationLocation` (str, opt), `deviceType` (str, opt), `roomType` (str, opt), `time` (str, opt), `meteorologicalType` (str, opt)
Weather & Environment	`CheckWeatherIndex`	Query location-specific weather indices by time and type. *Args:* `destinationLocation` (str, opt), `deviceType` (str, opt), `roomType` (str, opt), `time` (str, opt), `indexType` (str), `equipmentType` (str, opt)
Navigation & Travel	`TakeTaxi`	Handles taxi and ride-hailing booking, calling, and fare estimation. *Args:* `appName` (str, opt), `departureLocation` (str, opt), `destinationLocation` (str, opt), `taxiType` (str, opt), `phoneNumber` (str, opt), `passenger` (str, opt), `tag` (str, opt), `seatNumber` (str, opt), `time` (str, opt)
Clock & Calendar	`DeleteAlarm`	Removes alarms from list; excludes disabling or turning off. *Args:* `event` (str, opt), `rangeType` (str, opt), `state` (str, opt), `time` (str, opt), `index` (str, opt), `entityId` (str, opt), `deviceType` (str, opt)
Clock & Calendar	`CloseRing`	Mutes alarm audio without disabling the schedule. *Args:* None