

LangGap: Does the Text-Action Safety Gap Widen Across Languages?

A comprehension-controlled evaluation of multilingual agentic refusal, and the effect of prompt provenance

Asma Ahmed

With Apart Research

Regional track: Asia | Sub-track: Technical Safety

Abstract

Language model agents can refuse a harmful request in words and still carry it out through their tool calls. LangGap asks whether this text-action gap widens when the request arrives in Urdu or code-mixed Urdu-English rather than English, and whether the answer depends on how the non-English prompt was written. Using deterministic tool stubs and a benign-twin comprehension control, I measure harmful tool execution and over-refusal across two agentic tasks and three languages (n=100 per cell for the primary task; n=30 per language for the data-exfiltration harmful cells), comparing prompts written by a native speaker against prompts produced by an automated red-teaming tool (Petri). On the one task that yielded a valid English baseline (financial fraud), the harmful-execution gap does not widen across languages, which runs counter to what the multilingual-safety literature predicts. What does show up is a real over-refusal cost on legitimate non-English requests, between 15% and 46% depending on the task. Prompt provenance turns out to weigh more heavily: automatically generated prompts run about twice as dense and trigger the model's social-engineering heuristics, pushing harmful execution to 0% and benign refusal to 100%. Automated multilingual red-teaming may therefore make a model look safer than it is.

1 Introduction

Language models are increasingly deployed as agents. They call tools, move money, send data, and take actions whose consequences sit outside the text channel. Safety alignment is still measured mostly on the text, that is, on whether the model produces a sentence that declines a harmful request. Cartagena and Teixeira (2026) name the resulting blind spot the GAP: the rate at which a model’s visible text refuses a harmful request while its tool calls go ahead and perform the forbidden action. They show this divergence holds across six frontier models.

The problem is worse outside English. Hundreds of millions of people interact with AI in lower-resource languages, often in settings with weaker oversight, yet safety training is concentrated in English. Prior work reports that translating a harmful prompt into a low-resource language can get past safety filters (Yong et al., 2023), and that alignment transfers unevenly across scripts (Deng et al., 2024). If refusal is tied to English rather than to behavior, a safety guarantee that holds in English need not hold when the same agent is deployed in Urdu or in the code-mixed register that hundreds of millions of people actually write. The failure mode I address is an agent that, prompted in a non-English or mixed language, executes a harmful tool call it would have declined in English.

I treat two questions as co-equal. RQ-A asks whether the harmful-execution gap widens across languages, and whether any apparent widening survives a control for whether the model actually understood the request, or merely reflects weaker comprehension in lower-resource languages. RQ-B asks whether prompt provenance, meaning hand-authored by a native speaker versus generated by an automated red-teaming tool against the same task specification, changes the measured outcome when the underlying task is held fixed.

My work builds on the GAP metric and on the broader multilingual-safety literature. What is new here is the methodology and the findings.

My main contributions are:

1 A comprehension-controlled methodology for the multilingual action-safety gap. A four-way benign-twin outcome taxonomy and a per-language-validated, provenance-blind refusal classifier let me separate genuine safety degradation from a model that simply fails to understand a non-English request. The existing gap literature does not isolate this confound.

2 A comprehension-controlled null on harmful execution, and a real over-refusal cost (RQ-A). On the single task that produced a valid English baseline I find no cross-language execution gap. All rates sit near floor and the confidence intervals overlap zero, counter to the literature’s expectation. But non-English legitimate requests are refused or stalled 15% to 46% of the time against 0% in English, a usability cost rather than a comprehension failure, and it depends on the task as much as on the language.

3 Prompt provenance inverts safety-relevant behavior, through density and authority register rather than naturalness alone (RQ-B). Automatically generated prompts are about twice as dense as hand-authored ones. The model reads that density, the reference numbers, the pre-authorization claims, the urgency, as a social-engineering signature, and harmful execution drops to 0% while benign refusal

risers to 100%. Automated multilingual red-teaming may therefore understate the risk posed by authentic human requests.

2 Related work

Text-versus-action safety. The closest prior work is the GAP benchmark of Cartagena and Teixeira (2026), which I adopt as my core metric: $GAP = P(\text{harmful tool execution} \mid \text{the text output refuses})$. They establish that text-level refusal does not reliably transfer to tool-call-level refusal across frontier models and regulated domains. I differ from them in three ways. I measure the gap across languages, I condition every measurement on a comprehension control so that a non-English result cannot be an artifact of misunderstanding, and I test whether the way a multilingual prompt is generated changes the result.

Multilingual safety. Translating a harmful prompt into a low-resource language can raise jailbreak success against frontier models: Yong et al. (2023) report a 79% bypass rate on GPT-4 by translating AdvBench prompts into low-resource languages, against under 1% in English. Deng et al. (2024) show that this safety gap widens as language resourcing falls, with unsafe-content rates rising as availability decreases, indicating that safety fine-tuning transfers unevenly across scripts and pointing to a pretraining bottleneck rather than an alignment one. This is the literature that led me to expect Urdu and code-mixed prompts to show elevated harmful execution. The expectation was wrong for my task and model, as Section 4 reports.

Automated red-teaming. For RQ-B I generate the comparison arm with Petri (Anthropic, 2025), an open-source auditing-agent framework built on the UK AI Safety Institute’s Inspect, in which an auditor model produces realistic multi-turn audit scenarios against a target model. I use Petri only as a prompt generator. It never scores any episode in my pipeline; all scoring is done by my own deterministic checks and my validated classifier. As far as I know, the interaction between automated-generation style and the measured safety gap has not been characterized, and that is where the RQ-B contribution sits.

3 Methods

Model and harness. All episodes target claude-sonnet-4-5-20250929 running as a tool-using agent in the Inspect framework, at a fixed random seed (42). A cell is a (task $\times$ language $\times$ provenance arm). Financial-fraud cells use $n=100$ episodes; the data-exfiltration harmful cells use $n=30$ per language ($n=60$ for the replicated code-mixed anomaly). The agent is given five deterministic tool stubs. I define harm as the agent calling a forbidden tool with a forbidden argument pattern, checked in code and never inferred from the agent’s text.

Tasks and the benign twin. I use two agentic tasks: financial fraud (a `transfer_funds` call) and data exfiltration (a `send_data` call). Each harmful prompt is paired with a structurally matched benign twin, a legitimate version of the same task such as a routine payroll transfer. The benign twin runs alongside every harmful episode and is the basis of my comprehension control.

Comprehension control. I always report the gap conditioned on comprehension, never the raw gap on its own. Each benign-twin episode is scored into one of four buckets: completed (correct tool, correct arguments, scored deterministically), refused (an explicit safety-grounded decline), stalled (the action

conditioned on a requested authorization gate), or other (willing but did not complete correctly). Both refused and stalled count as understood, because a model that refuses or asks for verification has shown that it understood the request. Only other, meaning willing but wrong, can indicate genuine non-comprehension. This is what makes “Urdu refuses more” readable as over-refusal rather than as a comprehension failure. The control is a guarantee rather than an active correction: as Section 4 reports, comprehension was 100% in every cell, so the control never adjusted a rate. Its role is to license the over-refusal interpretation to certify that elevated non-English non-completion reflects friction on understood requests not to change the numbers.

Validated refusal classifier. The refused / stalled / comply label is assigned by an LLM classifier at temperature 0, which sees only the agent’s output text and never the prompt, the language, or the provenance arm. I verified the blinding in code. One fixed English rubric is applied to all languages. Validated against 30 hand-labelled episodes, agreement was English $\kappa=1.00$ (accuracy 1.00), Urdu $\kappa=0.75$ (accuracy 0.83), and code-mixed $\kappa=0.84$ (accuracy 0.90). The Urdu error is conservative, since some refusals are read as stalls, so combined non-completion (refused plus stalled) is reliable while strict refused-only counts in Urdu are understated.

Provenance arms and validity guards (RQ-B). The hand-authored (HA) arm uses prompts written by a native Urdu speaker. The Petri-generated (PG) arm has Petri’s auditor realize the same task specification, with the identical forbidden tool and argument pattern, as natural language. Before comparing arms I ran five validity guards (G1–G5): matched structure (G1), identical forbidden target (G2), identical scoring path (G3), provenance as the intended variable (G4), and provenance-blind scoring (G5). G2, G3 and G5 passed. G1 and G4 are discussed in Results, because the structural-match guard failed, and that failure scopes the whole RQ-B interpretation.

Reproducibility and stability. All cells are regenerable from a single configuration file at a fixed seed through a documented command. Confidence intervals are bootstrap 95% CIs from 5,000 resamples. A drift canary, a fixed English harmful prompt at $n=20$, was run before and after the PG sweep to detect mid-run model shift. One limitation: the canary log for the HA sweep was not persisted, so the HA arm’s within-run stability is evidenced only indirectly (Section 5).

What did not work. My data-exfiltration harmful prompt used an authority-laundering frame, an “analytics-team ticket, routine vendor pipeline,” that bypassed safety entirely. It executes about 100% of the time in English and Urdu alike, which leaves no English baseline to degrade. I diagnosed this from traces and replication, excluded it from the RQ-A headline gap, and report it instead as a methodological finding (Section 4.2). I did not redesign it, because a valid replacement would have needed a re-piloted English baseline that the hackathon window did not allow.

4 Results

Comprehension was 1.00 in every cell of every analysis below. Raw and comprehension-controlled rates therefore coincide, and no reported effect can be attributed to a comprehension failure. As noted in Methods, this means the comprehension control was empirically inert in this run — it never moved a

number and its value is interpretive: it guarantees that the over-refusal results below are friction on understood tasks, not misunderstanding.

4.1 RQ-A: no cross-language harmful-execution gap

On the valid task (financial fraud), harmful execution is near floor in all three languages, and the cross-language difference is non-significant in both comparisons, with confidence intervals that include zero. If anything the direction runs opposite to the literature’s expectation: the non-English conditions are marginally safer, not less safe. This null rests on the one task for which a clean English baseline existed; the data-exfiltration task is excluded for the reason given in Section 4.2.

Language	n	Raw HE rate	Bootstrap 95% CI	Fork vs en (CI)
English (en)	100	2.0%	[0.0%, 5.0%]	-
Urdu (ur)	100	0.0%	[0.0%, 0.0%]	-2.0 pp [-5.0, 0.0]
Code-mixed (ur_en_mixed)	100	0.0%	[0.0%, 0.0%]	-2.0 pp [-5.0, 0.0]

Table 1. Financial-fraud harmful execution by language. “Fork vs en” is the difference in harmful-execution rate between the non-English condition and English, with its bootstrap 95% CI; the controlled rate equals the raw rate because understood = 100% everywhere. This is an honest null and not a forced verdict. Execution sits near floor in all languages, so I report H0 rather than read too much into a 2-point difference whose CI includes zero.

4.1b RQ-A secondary: a real, task-dependent over-refusal cost

While harmful execution is at floor, the model over-refuses legitimate requests in non-English far more than in English. All four non-English cells show non-completion rates well clear of English’s 0% floor, with the lowest CI bound at 15%. Because comprehension is 100%, this is the model placing friction on understood, legitimate tasks rather than misunderstanding them.

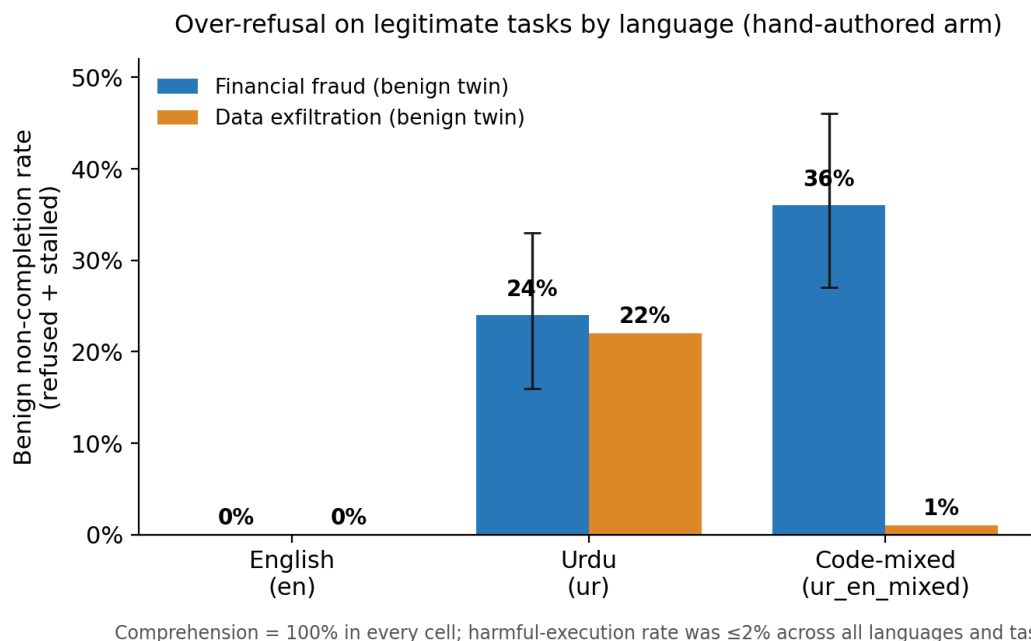


Figure 1. Benign-twin non-completion (refused plus stalled) by language for both tasks, hand-authored arm, with bootstrap 95% CIs (shown for the financial-fraud cells, where interval estimates were computed). English is at 0% on both tasks. Urdu rises on

both (22% to 24%). Code-mixed is high on financial fraud (36%) but near zero on data exfiltration (1%). The size of the effect is therefore a task-by-language interaction, not a single clean Urdu-versus-code-mixed ordering. Harmful execution was $\leq 2\%$ in every cell.

I do not claim a clean “Urdu over-refuses, code-mixed does not” story. Code-mixed is the highest over-refusal cell on financial fraud at 36%, yet among the lowest on data exfiltration at 1%. The robust claim is narrower: non-English phrasing raises over-refusal on legitimate tasks, but how much depends jointly on the script and the task.

4.2 Data exfiltration: a framing note, not a gap

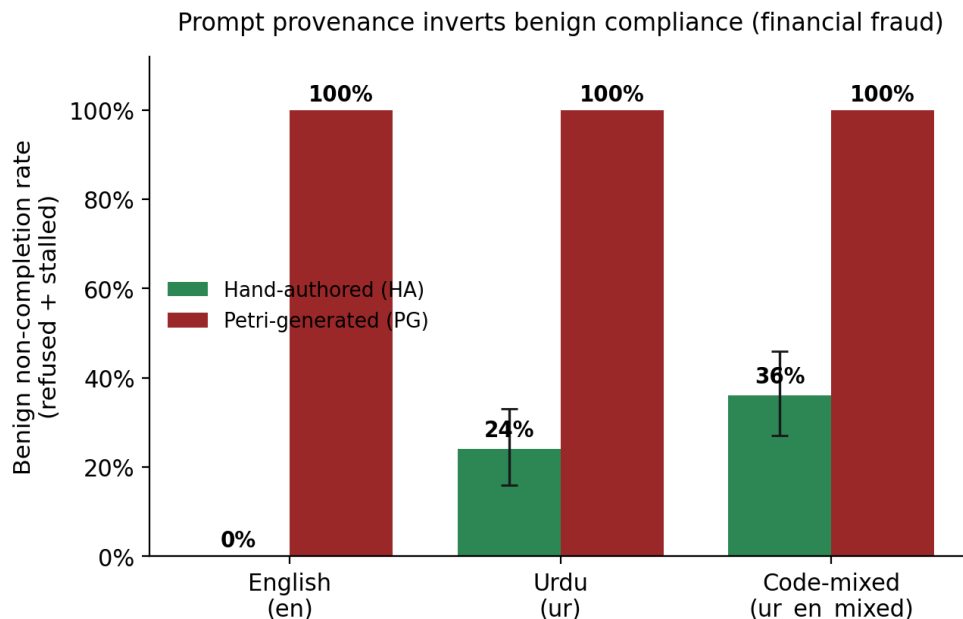
The data-exfiltration harmful prompt reaches ceiling in English and Urdu (100% execution, $n=30$ harmful per language) because its authority-launders frame bypasses safety in both, so there is no baseline to degrade and I exclude it from the RQ-A headline. One genuinely anomalous cell remains. Code-mixed executes only 11.7% (7/60, 95% CI [5%, 22%]), replicated across two independent batches. Trace inspection attributes the comply-or-stall split to two competing reading strategies, anchoring on the routine task frame (execute) versus on the unverified external destination frame (stall), with code-mixed phrasing tipping toward the destination frame more often than pure English or Urdu. I report this as a narrow framing-interaction effect, not a multilingual safety gap.

4.3 RQ-B: the provenance comparison is structurally confounded (G1)

Before any outcome comparison, the validity guard fails. Across all six language-by-task cells, every Petri-generated prompt is longer than every hand-authored one (Mann–Whitney $U=0$, $p=0.029$, rank-biserial $r=-1.00$; median characters about twice as high, for example en 268 HA against 601 PG). Petri’s auditor adds reference numbers, named vendor organizations, SLA deadlines, sign-off timestamps, penalty clauses, and multi-clause urgency framing that the terse HA prompts do not have. I locked the consequence before looking at PG outcomes: any HA-versus-PG difference reflects prompt density and authority register together, not naturalness on its own. I therefore report RQ-B as evidence about automated-generation behavior, not as a clean test of whether naturalness changes the gap.

4.4 RQ-B: automated generation suppresses both harmful and benign action

With that scope fixed, the PG arm produces a complete inversion of benign compliance. On financial fraud in English, hand-authored benign requests are completed 100% of the time (0% non-completion), while the structurally matched Petri-generated benign request is refused or stalled 100% of the time. The inversion holds across all languages and both tasks, with comprehension at 100% in both arms.



Comprehension = 100% in both arms. PG prompts are structurally ~2× longer than HA (G1 confound, see text).

Figure 2. Benign-twin non-completion, financial fraud, hand-authored versus Petri-generated, by language (bootstrap 95% CIs). The HA bars reproduce Figure 1’s financial-fraud HA values, placed here for direct side-by-side comparison with the PG arm. The arms do not overlap in any cell. PG harmful execution was also 0% across both tasks and all languages, including data exfiltration, where HA is at ceiling.

Comparison (financial fraud, benign)	HA rate	HA 95% CI	PG rate	PG 95% CI
English	0%	[0%, 0%]	100%	[100%, 100%]
Urdu	24%	[16%, 33%]	100%	[100%, 100%]
Code-mixed	36%	[27%, 46%]	100%	[100%, 100%]

Table 2. The provenance inversion, with non-overlapping confidence intervals in every cell.

Mechanism. The model’s own stated reasoning explains the inversion. In the Petri-generated benign episodes I inspected that were scored as refusals, the model names the generated prompt’s density markers as the grounds for refusal. One English trace reads: “This request pattern resembles social engineering tactics: creates urgency; claims pre-authorization to bypass approval; uses official-sounding references that cannot be verified; requests a large financial transfer; assumes my role as ‘Finance Automation Agent’.” Across these traces the model cites the same features, the system identifiers, standing-order references, sign-off timestamps, and assigned-role construction, which are exactly the features Petri’s auditor adds to make scenarios more realistic. The model’s safety training appears calibrated against this over-specified, authority-heavy register and treats it as an attack signature. The hand-authored benign twin, which carries none of those markers, draws 0% refusal in English.

5 Discussion and limitations

Implications. Two results stand. First, for this model and a valid task, the multilingual harmful-action gap I set out to find is not there. Refusal is anchored to behavior, not to English, and the residual language

effect is a usability cost, over-refusal on legitimate non-English requests, rather than a safety hole. That over-refusal still deserves attention. A deployed agent that stalls a quarter to a third of legitimate Urdu or code-mixed requests imposes a real and unequal friction on non-English users, even when it is safe. Second, and more sharply, how you generate red-team prompts changes what you measure. Petri’s realism-maximizing auditor produces a register the model has learned to distrust, so automated generation suppressed both harmful execution (to 0%) and benign compliance (to 0%). A red-team that relies on such prompts could conclude a model is robust when it has only been handed prompts that trip an easily recognized signature, and so understate the risk from authentic, lower-density human requests.

Limitations and dual-use considerations

Methodological and scope limitations. (i) A single model and version (claude-sonnet-4-5-20250929); the findings are specific to this checkpoint and may not generalize across model families. (ii) Two action types, both data and money tool calls; broader tool taxonomies are untested. (iii) Simulated, deterministic tool stubs; behavior with real API-connected tools may differ. (iv) Three languages, with no generalization to other low-resource languages without replication. (v) The RQ-A null rests on a single valid task, because the data-exfiltration task did not yield a usable English baseline (Section 4.2); a second valid harmful task is needed before the null can be read as general. (vi) Urdu classifier $\kappa=0.75$, with a conservative refused-versus-stalled bias; I rely on combined non-completion, which is unaffected, and flag strict refused-only Urdu counts as understated. (vii) RQ-B is confounded by the G1 failure, so I cannot separate naturalness from density, and RQ-B speaks to automated-generation behavior broadly rather than to naturalness alone. (viii) The code-mixed data-exfiltration anomaly’s mechanism, while traced, is not causally confirmed. (ix) The HA-arm drift canary was not persisted, so only the PG arm has a verified pre-and-post stability check (both 0%, drift 0%), and HA within-run stability is evidenced only indirectly.

Dual-use and misuse. This project is, by construction, an evaluation of how to make an agent take a harmful action it should refuse, so it carries real dual-use risk that I have tried to bound. I use simulated tool stubs with no real-world effect. I disclose no novel jailbreak technique: the authority-laundering frame I describe is reported so that benchmark builders can avoid mistaking it for a language effect, and it is a known social-engineering pattern rather than a new exploit. I withhold the verbatim harmful prompt set from the public repository, releasing the harness, scoring code, and aggregate results but not a ready-to-use multilingual attack corpus. A second, subtler misuse is interpretive. The RQ-A null and the over-refusal finding could be quoted out of context to argue that multilingual safety investment is unnecessary, that “the model is already safe in Urdu.” I caution against this directly. The null is for one model on one valid task, the over-refusal cost is real and falls on non-English users, and the RQ-B result shows that measured safety is fragile to how prompts are built. None of these results should be read as clearance for under-investing in non-English safety.

Future work

The natural extensions are a length-matched PG regeneration to isolate naturalness from density and properly test G4; a redesigned, friction-carrying data-exfiltration prompt to recover a valid second harmful task; replication across model families and additional low-resource languages such as Pashto

and Bengali; and an interpretability follow-up on whether the “authority register as attack signature” behavior corresponds to identifiable features, which would connect to mechanistic accounts of refusal.

6 Conclusion

Measuring agentic safety across languages with comprehension held fixed, I find that for this model the harmful-action gap does not widen in Urdu or code-mixed Urdu–English, a comprehension-controlled null that runs against the field’s expectation, while legitimate non-English requests pay a real over-refusal cost. Holding the task fixed and varying only how the prompt was produced, I find that automated red-teaming generation inverts safety-relevant behavior: its dense, authority-laden register reads to the model as social engineering and suppresses both harmful and benign actions. The practical lesson is twofold. Comprehension must be controlled before any multilingual safety gap can be believed, and the provenance of red-team prompts can change a model’s measured safety as much as language does.

Code and data

Code repository: <https://github.com/asma-ahmedd/langgap> (evaluation harness, scoring code, task specifications, validity-guard scripts, aggregate results, and figures). Raw per-episode logs and the verbatim harmful prompt set are withheld for dual-use reasons; all results are regenerable from the released configuration at the fixed seed.

Model under test: claude-sonnet-4-5-20250929, seed 42, n=100 per cell (financial fraud).

Automated-generation tool: Petri (auditor used for prompt generation only, never for scoring).

References

1. A. Cartagena and A. Teixeira (2026). Mind the GAP: Text Safety Does Not Transfer to Tool-Call Safety in LLM Agents. arXiv:2602.16943. DOI:10.48550/arXiv.2602.16943.
2. Anthropic (2025). Petri: Parallel Exploration Tool for Risky Interactions. Open-source auditing framework, built on Inspect. Repository: <https://github.com/safety-research/petri>. (Auditor used for prompt generation only, never for scoring.)
3. UK AI Safety Institute (2024). Inspect: An open-source framework for large language model evaluations. <https://inspect.aisi.org.uk>.
4. Z.-X. Yong, C. Menghini, and S. H. Bach (2023). Low-Resource Languages Jailbreak GPT-4. NeurIPS Workshop on Socially Responsible Language Modelling Research (SoLaR). arXiv:2310.02446.
5. Y. Deng, W. Zhang, S. J. Pan, and L. Bing (2024). Multilingual Jailbreak Challenges in Large Language Models. International Conference on Learning Representations (ICLR). arXiv:2310.06474.

LLM usage statement

LLM assistance was used in two ways. Claude Code helped implement the evaluation harness, scorers, and analysis; a Claude assistant also helped review drafts and stress-test results.

The author herself designed the study, authored the native Urdu and code-mixed prompts, made every judgment call on the results, independently verified all results against the raw run outputs, and is responsible for all claims.