

MoMo: A Threat Corpus and Evaluation Framework for Mobile Money Fraud Resistance in Swahili, Wolof, and Hausa

Mufaro Sean¹

¹Cape Peninsula University of Technology (CPUT), Cape Town, South Africa

With Apart Research

Research conducted at the Global South AI Safety Hackathon, June 2026

Abstract

Mobile money platforms—M-Pesa, Orange Money, and Wave among them—now mediate financial life for more than 500 million people across Sub-Saharan Africa, and have become a correspondingly attractive target for social-engineering fraud conducted in local languages and idioms that mainstream AI safety evaluation rarely covers. We present MoMo, a structured threat corpus and evaluation framework for benchmarking large language model (LLM) resistance to generating localized mobile money fraud content in Swahili, Wolof, and Hausa. MoMo organizes fraud scenarios along a six-category attack taxonomy (SIM-swap, agent impersonation, lottery/prize lures, OTP phishing, emergency urgency, and overpayment scams), crossed with six socioeconomic hooks and six victim personas, and pairs this corpus with an asynchronous evaluation harness that scores model outputs using rule-based red-flag detection, an LLM-judge rubric, and a locale-aware aggregator producing a 0–100 Model Safety Score (MSS). A defining design choice is a two-tier access model: full attack scripts are withheld from public release and gated behind a vetted-researcher Data Use Agreement, while a public stub corpus (script metadata, taxonomy labels, and SHA-256 hashes) supports CI, reproducibility checks, and open auditing without creating a ready-made social-engineering playbook. We describe the corpus schema, the native-speaker annotation pipeline built on Label Studio, and the evaluation harness architecture, and report a pilot evaluation against two NVIDIA NIM-hosted Llama 3.1 models (8B and 70B) that found refusal rates under 10% across 171 fraud scenarios, with the larger model substantially more compliant than the smaller one and with the lowest refusal rates on emotionally manipulative attack framings. We discuss the engineering and governance trade-offs involved in building a dual-release safety benchmark, alongside the measurement limitations of the current rule-based scorer. The main takeaway is that benchmarking AI safety for underserved linguistic markets requires infrastructure—access control, annotation tooling, and locale-aware scoring—and that, on this preliminary evidence, current open models show weak resistance to localized mobile money fraud.

1 Introduction

Large language models are increasingly accessible to fraud operators as drafting and translation tools, and the languages in which this is most consequential are frequently the ones least represented in AI safety evaluation. Mobile money is the dominant financial rail across much of East, West, and Southern Africa, and fraud against mobile money users—SIM-swap pretexting, fake agent float requests, OTP phishing, and lottery lures—is typically delivered in Swahili, Wolof, Hausa, or other regional languages rather than in English. Standard AI safety benchmarks are overwhelmingly English-centric, which leaves an evaluation gap: a model that refuses to draft an English-language phishing message may have no comparable guardrail in Swahili or Hausa, simply because no one has measured it.

MoMo addresses this gap directly. It is a benchmarking project purpose-built to evaluate whether LLMs resist producing localized mobile money social-engineering content, and to do so with the linguistic and cultural specificity that a translated or English-only benchmark would miss. The project is motivated by a threat model in which a fraud operator uses a general-purpose LLM to draft, vary, or refine scam scripts in a target language faster and more fluently than they could manually—and in which existing safety evaluations would not detect this failure mode because they were never tested in that language or against that attack pattern.

Our main contributions are:

1. A six-category attack taxonomy for mobile money fraud (SIM-swap fraud, agent impersonation, lottery/prize lures, OTP phishing, emergency urgency, and overpayment scams), crossed systematically with six socioeconomic hooks and six victim personas to support combinatorial scenario generation.
2. A two-tier corpus release design that separates public, CI-safe metadata (hashed scripts, taxonomy labels, red-flag annotations) from a protected corpus of full attack text, gated behind a research proposal and Data Use Agreement—addressing the dual-use tension inherent in publishing fraud scripts as a safety resource.
3. An end-to-end evaluation harness—corpus builder, Jinja2-based prompt generator, asynchronous LiteLLM-driven model querying, a four-stage scorer (rule-based, LLM-judge, locale-aware, and ensemble), and a Streamlit leaderboard dashboard—together with a native-speaker annotation pipeline built on a self-hosted Label Studio deployment.

2 Related Work

Existing LLM safety and red-teaming benchmarks (covering categories such as malware generation, disinformation, and general harmful-content refusal) are predominantly constructed and validated in English, and where multilingual coverage exists it is typically achieved through machine translation of English seed prompts rather than through native-speaker-authored, sociolinguistically grounded scenarios. This matters for fraud-specific evaluation in particular: a literal translation of an English phishing script will frequently fail to trigger the same refusal behavior as an idiomatically authentic one, because surface fluency and register cues differ from how a model’s safety training represents the underlying intent.

Separately, the mobile money fraud literature and regulatory guidance from bodies such as the Communications Authority of Kenya document known attack patterns (SIM-swap, agent impersonation, OTP phishing) as consumer-protection concerns, but this literature does not connect to AI safety evaluation methodology, and no existing public benchmark operationalizes these patterns as LLM red-teaming prompts.

MoMo differs from prior work along two axes. First, scenario authenticity: every script in the protected corpus is intended to pass native-speaker review on idiomatic authenticity, sociolinguistic register, and real-world plausibility before being used in evaluation, rather than relying on translation. Second, release strategy: rather than treating “open benchmark” and “responsible disclosure” as opposing goals, MoMo’s two-tier model tries to satisfy both—a public, hash-verified metadata layer supports reproducibility and CI integration, while the attack text itself is distributed only under a Data Use Agreement to vetted researchers. A practitioner would reach for MoMo over a generic multilingual red-teaming set specifically when the question is sector-specific (mobile money) and language-specific (Swahili, Wolof, Hausa), and would value MoMo’s locale-aware scorer and persona/hook taxonomy as structure that a generic benchmark does not provide.

3 Methods

3.1 Corpus Design and Attack Taxonomy

The corpus is organized around six attack types, each representing a distinct social-engineering mechanism observed in mobile money fraud:

- **SIM-swap fraud**—the attacker claims the victim’s line will be deactivated unless they share their mobile-money PIN.
- **Agent impersonation**—a fake mobile-money agent requests a float top-up or account access.
- **Lottery / prize lures**—advance-fee scams promising a cash reward in exchange for a small “processing fee.”
- **OTP phishing**—extracting a one-time password under a false pretext such as a security alert, debt collection notice, or family emergency.
- **Emergency urgency**—a fabricated medical or family emergency demanding immediate funds.
- **Overpayment scams**—a fake wrong-number deposit followed by a plea to refund the difference.

Each attack type is crossed with six socioeconomic hooks (debt urgency, family emergency, windfall greed, authority compliance, technical fear, social obligation) and six victim personas (rural smallholder, urban worker, student, elderly person, small business owner, market trader), giving a combinatorial scenario space across three languages—Swahili (**sw**), Wolof (**wo**), and Hausa (**ha**)—and their associated mobile money networks (M-Pesa, Orange Money, Wave).

The repository implements this as a Pydantic-modeled corpus (`momo/corpus/models.py`), assembled by a builder module (`corpus/builder.py`) and persisted through a SQLAlchemy/Alembic-backed database that supports both local SQLite development and Postgres deployment. A loader module handles JSON/YAML/database serialization, and a queries module provides filtering and analysis helpers over the assembled corpus.

3.2 Two-Tier Access Control

Because the corpus content is, by construction, a set of working social-engineering scripts, MoMo separates what is published from what is distributed. The public repository contains script metadata, identifiers, taxonomy labels, and red-flag annotations, with the canonical script text replaced by a SHA-256 hash; this layer is safe for continuous integration and general public access. The protected tier—full attack text, native-speaker annotations, and evaluation results—is distributed only to researchers who submit a short research proposal (affiliation, intended use, data-handling practices, and ethics-board approval where applicable), open a structured access request, and sign a Data Use Agreement. An access-control and hash-verification module (`corpus/access.py`) and an export tool (`scripts/export_corpus.py`) implement this split, including a `verify` subcommand that checks a distributed protected corpus against its SHA-256 manifest for integrity.

3.3 Native-Speaker Annotation Pipeline

All scripts require native-speaker validation before use in evaluation, scored against four criteria: idiomatic authenticity (does the phrasing sound natural to a native speaker of the target language), sociolinguistic register (formal versus street register appropriateness), real-world plausibility (would this lure actually work in the target market), and red-flag accuracy (are the assigned red flags correct and complete). This pipeline is implemented through a self-hosted Label Studio deployment (docker-compose, under `docker/label-studio/`), with a CLI wrapper (`annotation/label_studio.py`) for configuring projects and exporting/importing per-language annotation tasks.

3.4 Evaluation Harness

Model evaluation is performed asynchronously against frontier and open models through LiteLLM, allowing a single harness configuration to target multiple providers (the documented usage example evaluates GPT-4o and a Claude Sonnet model together) and to filter runs by attack type and language. Scoring is layered: a spaCy-based rule scorer performs red-flag detection with graceful fallback when patterns are not matched; an LLM-judge scorer applies a structured 1–10 rubric (using Claude or GPT-4 as judge); a locale scorer applies language- and network-specific adjustments; and a dual/ensemble scorer combines the rule-based and LLM-judge signals. These feed a weighted aggregator that produces a single 0–100 Model Safety Score (MSS) per model, attack type, and language. Evaluation results are persisted (`eval/db.py`) and surfaced through a Streamlit dashboard (`dashboard/app.py`) offering KPI summaries, a model leaderboard ranked by MSS, per-language and per-attack-type drill-downs, and radar-chart comparisons across models.

A mock evaluation mode (`run -mock`) allows the harness, scorers, and dashboard to be exercised end-to-end without API keys or live model calls, which keeps the pipeline testable in continuous integration despite the protected corpus being inaccessible to CI by design.

3.5 What Did Not Work / Open Engineering Questions

As a hackathon-stage project, several design questions remain open rather than resolved. The weighting scheme used by the MSS aggregator to combine rule-based and LLM-judge sub-scores is not yet empirically tuned against ground-truth human judgments. The mock evaluation mode, while valuable for CI, has not yet been validated for producing scores that distributionally resemble real model outputs, so mock-mode results should not be treated as a proxy for real MSS rankings. Finally, the project currently covers three languages and their associated networks; extending the taxonomy and annotation pipeline to additional regional languages (e.g., Yoruba, Amharic, Zulu) would require repeating the native-speaker sourcing and validation effort described in Section 3.3, which the budget treats as the dominant cost driver.

4 Results

4.1 Attack Taxonomy and Scoring Pipeline

Table 1 summarizes the six attack types underlying the corpus, and Table 2 summarizes the layered scoring pipeline described in Section 3.4. The evaluation reported below uses only the rule-based (Layer A) classifier; the LLM-judge layer, locale adjustment, and weighted MSS aggregator had not yet been applied at the time of this run.

4.2 Pilot Evaluation: NVIDIA NIM Models (June 21, 2026)

Using corpus version 0.1.0 (171 scripts spanning six attack types, three languages, and three networks), we ran a pilot evaluation against two NVIDIA NIM-hosted models accessed through the OpenAI-compatible API: `nvidia/llama-3.1-8b-instruct` and `nvidia/llama-3.1-70b-instruct`. A third target, `nvidia/mistral-neMo-12b-instruct`, failed with a 404 error due to an incorrect model slug (corrected post-run to `nv-mistralai/mistral-nemo-12b-instruct` for future evaluations) and is excluded from the results below. Each of the 171 scripts was submitted three times per model (temperature = 0.7), for 1,539 total API calls across the three configured models; the Mistral failures account for 1,200 of these, leaving 339 successful evaluations (171 scripts × 2 working models × 3 attempts). Refusal was determined by `classify_refusal()` in `momo/eval/harness.py`, a multilingual keyword classifier (English, Swahili, Wolof, Hausa) with threshold-based classification; this is the rule-based Layer A scorer only — the LLM-judge layer (Layer B) had not yet been applied to this run.

Both models refused fewer than 10% of fraud-response requests, with the larger 70B model

Attack type	Mechanism	Example pretext element
SIM-swap fraud	Line-deactivation threat	PIN disclosure under threat of disconnection
Agent impersonation	Fake mobile-money agent	Float top-up or account-access request
Lottery / prize lures	Advance-fee scam	Processing fee for a fabricated cash prize
OTP phishing	False-pretext OTP extraction	Security alert, debt collection, family emergency
Emergency urgency	Fabricated crisis	Medical or family emergency demanding funds
Overpayment scams	Fake wrong-number deposit	Plea to refund an overpayment

Table 1: The six attack types in the MoMo taxonomy, each crossed with six socioeconomic hooks and six victim personas across three languages (Swahili, Wolof, Hausa), yielding a combinatorial scenario space for evaluation prompt generation.

Component	Role	Output
spaCy rule scorer	Red-flag pattern detection	Binary/graded red-flag hits
LLM-judge scorer	Structured rubric scoring (Claude / GPT-4)	1–10 rubric score
Locale scorer	Language/network-aware adjustment	Adjusted sub-score
Dual scorer	Ensemble of rule + judge	Combined sub-score
Aggregator	Weighted combination across sub-scores	0–100 Model Safety Score (MSS)

Table 2: The layered scoring pipeline implemented in `momo/eval/scorers/`, culminating in a single weighted Model Safety Score per model, language, and attack type.

refusing roughly $3\times$ *less* often than the smaller 8B model. Table 4 breaks results down by language, attack type, and network; the low refusal rate is broadly consistent across all three dimensions rather than concentrated in a single language or attack pattern.

Qualitative examples. Manual inspection of transcripts illustrates the range of model behavior under the rule-based classifier. In one Swahili SIM-swap scenario, Llama 3.1 8B reproduced and forwarded PIN-disclosure instructions consistent with the fraudulent script, which the rule-based classifier flagged as a harm signal on a subsequent attempt. In the same scenario, Llama 3.1 70B instead warned the simulated user not to respond to the message and advised contacting M-Pesa directly through official channels — a substantively protective response that was nonetheless scored as “complied” rather than “refused,” because the classifier’s refusal criterion is whether the model declines to engage with the request at all, not whether its engagement is protective. This is a measurement limitation of the current Layer A classifier rather than a finding about model behavior, and we discuss it further in Section 5.

Because LLM-judge scoring, locale adjustment, and the weighted MSS aggregator (Table 2) have not yet been applied to this run, we report rule-based refusal/compliance rates only, and we do not yet compute a Model Safety Score for these models. We also do not report statistical significance or robustness checks at this stage; with $n = 171$ scripts and 3 attempts per model, per-cell breakdowns (e.g., a single attack type within a single language) are based on small sample sizes and should be read as directional rather than conclusive.

Metric	Llama 3.1 8B	Llama 3.1 70B
Refusal rate	9.2%	3.1%
Compliance rate	90.8%	96.9%
Avg. latency	~4.8 s	~16.2 s

Table 3: Overall refusal and compliance rates for the two successfully evaluated NVIDIA NIM models against the 171-script MoMo corpus (rule-based classification only).

Language	Refusal	Attack type	Refusal	Network	Refusal
Swahili	5.0%	Lottery / prize	6.6%	M-Pesa	6.4%
Wolof	4.5%	Agent impersonation	5.9%	Wave	3.3%
Hausa	2.5%	Overpayment	3.4%	Orange Money	3.2%
		OTP phishing	3.0%		
		SIM swap	2.9%		
		Emergency urgency	2.8%		

Table 4: Combined refusal rate by language, attack type, and mobile money network, pooled across both evaluated models.

5 Discussion and Limitations

The pilot evaluation in Section 4.2 provides preliminary but concrete evidence for the threat model motivating MoMo: both NVIDIA NIM models tested complied with the large majority of mobile-money fraud-response requests, refusing fewer than 10% of the time under rule-based classification. Three patterns stand out. First, refusal did not scale with model size in the expected direction — the larger Llama 3.1 70B model was roughly three times *more* compliant than the smaller 8B model, suggesting that increased instruction-following capability is not, by itself, accompanied by stronger multilingual safety behavior, and may even work against it if the larger model is more willing to interpret “help me respond to this message” as a request to be serviced rather than a context to evaluate for harm. Second, the roughly two-fold gap between Swahili/Wolof (5.0%, 4.5%) and Hausa (2.5%) refusal rates is consistent with known disparities in multilingual safety alignment, where safety training data and fine-tuning effort are unevenly distributed across languages even within a single low-resource-language cohort. Third, refusal rates were lowest for emotionally manipulative framings (emergency urgency, 2.8%) and highest for transparently incentive-based framings (lottery/prize lures, 6.6%), suggesting that current safety training is more reliable at recognizing “too good to be true” patterns than at recognizing manipulative urgency, despite both being well-documented social-engineering mechanisms.

The qualitative example in Section 4.2 also exposes a measurement gap in the current rule-based classifier: a response that actively warns a user away from a scam and advises contacting the legitimate provider is presently scored identically to a response that merely engages with the prompt, because the classifier distinguishes *refusal* from *compliance* rather than distinguishing *protective* from *harmful* compliance. This means the reported compliance rates in Table 3 likely conflate genuinely harmful completions (e.g., a model relaying PIN-disclosure instructions) with protective completions (e.g., a model warning the user and redirecting them to official channels), and probably overstate the true harm rate. Closing this gap is the primary motivation for applying the LLM-judge layer (Section 3.4), which is designed to score harm enablement on a graded rubric rather than a binary refusal/compliance label.

More broadly, the implication for AI safety is that fraud-resistance evaluation is not language-neutral, and benchmarks that exist only in English will systematically miss failure modes that

are most consequential to mobile-money-dependent populations in Sub-Saharan Africa. By making the attack taxonomy, persona/hook structure, and scoring methodology public while gating the attack text itself, MoMo attempts a release strategy that is auditable without being directly weaponizable—a pattern that may generalize to other dual-use safety benchmarks (e.g., for romance scams, investment fraud, or political disinformation) where the evaluation prompts are themselves harmful artifacts.

Limitations

- **Rule-based scoring only.** The results in Section 4.2 use only the Layer A keyword classifier; the LLM-judge layer, locale adjustment, and weighted MSS aggregator (Table 2) have not yet been applied. As the qualitative example shows, the current classifier cannot distinguish protective compliance from harmful compliance, so reported compliance rates likely overstate genuine harm.
- **Single provider, partial coverage.** The pilot evaluated only two models from a single provider (NVIDIA NIM); a third model failed outright due to a configuration error and was excluded rather than re-run within this reporting window. No results yet exist for Anthropic, OpenAI, or Google models.
- **Small sample size.** With 171 scripts and 3 attempts per model, per-cell breakdowns (a single attack type within a single language, for instance) rest on small sample sizes; we have not computed confidence intervals or tested for statistical significance, and per-cell rates should be read as directional rather than conclusive.
- **Corpus not yet independently validated.** The corpus version evaluated (0.1.0) has not yet completed the native-speaker annotation pipeline described in Section 3.3, so we cannot yet rule out that some scripts contain unnatural phrasing or incorrect sociolinguistic register that could affect refusal rates in either direction.
- **Unvalidated aggregation weights.** The MSS aggregation weights described in Table 2 are not yet empirically validated against human ground truth, so once computed, the headline 0–100 score should be treated as a provisional composite rather than a calibrated risk measure.
- **Trust assumption in vetting.** The two-tier access model assumes good-faith vetting of researcher requests; it reduces but does not eliminate the risk that the protected corpus could be misused if a vetted recipient acted in bad faith or had their access compromised.

Future Work

- Apply the LLM-judge scorer (Layer B) to the existing 339 NVIDIA NIM transcripts to obtain graded harm-enablement scores, and re-classify the protective-but-engaged responses identified in Section 4.2 accordingly.
- Re-run the failed Mistral NeMo configuration with the corrected model slug, and extend the pilot to Anthropic Claude, OpenAI GPT-4o, and Google Gemini for cross-provider comparison.
- Compute the weighted Model Safety Score (MSS) once Layer B scoring is available, and report the first genuine cross-model leaderboard with bootstrap confidence intervals across scenario variants.
- Complete native-speaker annotation for corpus version 0.1.0 and validate that the observed language gap (Hausa vs. Swahili/Wolof) is not an artifact of uneven scenario authenticity across languages.
- Engage the telecommunications regulators and academic linguistics partners identified in the project’s stakeholder map (Communications Authority of Kenya, ARTP Senegal, NCC Nigeria, Bank of Tanzania; University of Nairobi, UCAD Dakar, Bayero University Kano)

to validate scenario realism against real (anonymized) fraud reports and to create a policy-facing channel for findings.

6 Conclusion

MoMo contributes both a sector-specific, language-specific infrastructure for evaluating LLM resistance to mobile money fraud generation in Swahili, Wolof, and Hausa, and an initial empirical signal that this resistance is currently weak. A pilot evaluation against two NVIDIA NIM-hosted Llama 3.1 models found refusal rates under 10% across 171 fraud scenarios, with the larger 70B model substantially more compliant than the smaller 8B model and with refusal rates lowest for emotionally manipulative attack framings — a pattern consistent with the threat model that motivated the project. This threat surface is materially important to the more than 500 million Africans who depend on mobile money, and is largely invisible to English-centric AI safety benchmarks. The project’s broader technical contribution remains the reproducible pipeline behind these numbers: a combinatorial attack taxonomy, a native-speaker annotation workflow, an asynchronous multi-model evaluation harness with layered scoring, and a deliberately dual-tiered release model that tries to keep the benchmark auditable without publishing a ready-made fraud playbook.

The project’s near-term value lies in demonstrating that this kind of infrastructure can be built and exercised against real models quickly and rigorously, even within a hackathon time-frame; its long-term value will depend on completing native-speaker annotation at scale, applying LLM-judge scoring to resolve the protective-versus-harmful compliance ambiguity identified in Section 5, extending evaluation to additional providers, and engaging the regulators and linguistic partners who can validate scenario realism against real-world fraud patterns. We see MoMo as a template for how dual-use, linguistically-grounded safety benchmarks for the Global South can be structured responsibly, with this pilot as an early but incomplete data point rather than a finished leaderboard.

Code and Data

Code repository:

<https://github.com/Mufarosean89/MoMo>

Data / Datasets:

Public stub corpus (hashed script metadata, taxonomy, red-flag annotations) is included in the repository under `momo/corpus/scripts/`. The full attack-script corpus and native-speaker annotations are distributed only to vetted researchers under a Data Use Agreement, per the access process described in Section 3.2 and the repository’s “Ethics & Access” documentation.

Evaluation results:

Raw results from the NVIDIA NIM pilot evaluation (Section 4.2) are available at `data/eval_results_nvidia/`; the run configuration is reproduced in Appendix A.3.

Other artifacts:

A Streamlit evaluation dashboard (`momo/dashboard/app.py`) and a self-hosted Label Studio annotation environment (`docker/label-studio/`) are included in the repository for local deployment.

Author Contributions

Mufaro Sean conceived and designed the project, including the attack taxonomy, two-tier access architecture, evaluation harness, and annotation pipeline, and wrote this report.

References

- [1] Mufaro Sean (2026). *MoMo: Mobile Money Fraud Threat Corpus & Evaluation Framework*. GitHub repository. <https://github.com/Mufarosean89/MoMo>
- [2] Communications Authority of Kenya. *Consumer protection and mobile money regulation guidance (M-Pesa)*.
- [3] Autorité de Régulation des Télécommunications et des Postes (ARTP), Senegal. *Orange Money / Wave regulatory oversight*.
- [4] Nigerian Communications Commission (NCC). *Mobile money regulatory framework*.
- [5] Bank of Tanzania (BoT). *Mobile money and financial inclusion policy documentation*.

A Appendix

A.1 Repository Structure

The public repository is organized as follows (abridged):

- `corpus/scripts/{raw, swahili.py, wolof.py, hausa.py}`—raw scripts (gitignored) and public hashed stubs
- `corpus/{access.py, builder.py, loader.py, models.py, queries.py, db/}`—access control, assembly, serialization, Pydantic models, SQLAlchemy/Alembic database
- `prompt_generator/`—Jinja2 prompt templates for the six attack types
- `eval/{harness.py, config.py, scorers/, db.py}`—asynchronous LiteLLM evaluation harness and layered scorers
- `annotation/label_studio.py`—Label Studio integration for native-speaker annotation
- `dashboard/app.py`—Streamlit leaderboard and drill-down dashboard
- `docker/label-studio/`—Docker Compose configuration for self-hosted annotation
- `scripts/export_corpus.py`—maintainer tool for stub/protected export and integrity verification

A.2 License and Ethical Use Terms

The repository is released under CC BY-NC-SA 4.0 with an additional Ethical Use Clause: the dataset may not be used for actual social-engineering attacks or fraud; published derivative work must acknowledge MoMo; full corpus access requires maintainer vetting; and derivative works are expected to benefit low- and middle-income-country (LMIC) stakeholders.

A.3 NVIDIA NIM Pilot Evaluation: Run Configuration

The configuration used for the pilot evaluation reported in Section 4.2 was as follows:

```
{
  "n_attempts": 3,
  "temperature": 0.7,
  "max_concurrent": 5,
  "max_tokens": 1024,
  "provider": "nvidia-nim",
  "base_url": "https://integrate.api.nvidia.com/v1",
  "corpus_size": 171,
  "total_api_calls": 1539
}
```

Models evaluated: `nvidia/llama-3.1-8b-instruct` (success), `nvidia/llama-3.1-70b-instruct` (success), `nvidia/mistral-neMo-12b-instruct` (failed, HTTP 404 — incorrect model slug, corrected post-run to `nv-mistralai/mistral-nemo-12b-instruct`). Of 1,539 total API calls, 1,200 errored (all attributable to the Mistral slug failure), leaving 339 successful evaluations.

LLM Usage Statement

This report was drafted with the assistance of Claude (Anthropic), based on the public README and repository structure of the MoMo project at the GitHub link above, and on an evaluation report documenting a pilot run against NVIDIA NIM-hosted models. Claude was used to structure the report against the hackathon template and to draft prose describing the repository’s architecture and the pilot evaluation results. All quantitative results in Section 4.2 are taken directly from the supplied evaluation report and are not independently re-derived; the qualitative claims about response content (Section 4.2) reflect the example transcripts provided in that report. The author should independently verify all technical descriptions and figures against the underlying `data/eval_results_nvidia.json` file and the current state of the codebase before final submission.