BantuNomics NUMBERS Fine-Tuning

Technical Report — v1.1

Does machine-verified Bantu numeracy data improve language models?

3MegaLabs · BantuNomics

2026-08-09 · CONFIDENTIAL

Fine-Tuning Frontier-Class Small Models on BantuNomics Structured Numeracy Data

Technical Report — v1.0 · 2026-08-08

Program: NUMBERS fine-tuning (Bantu Calculator data → LLM capability) **Author:** 3MegaLabs / BantuNomics · **Compute:** Modal (A100-40GB) · **Total spend:** ≈ $41 **Repository:** `mmmwembe/3mega-bantunomics-numbers-finetuning` (private)

Executive summary

We tested whether BantuNomics' machine-verified Bantu numeracy data measurably improves language models, using the strictest available protocol: exact-match generation on frozen, contamination-checked, held-out test sets, with a matched **control condition** to isolate the contribution of data *content* from the act of fine-tuning itself.

Headline results:

Round	Model	Baseline	Fine-tuned	Control (shuffled targets)
1	Qwen 3.5-4B-Base	3.6%	**56.8%** (16×)	—
1	Gemma 4 E4B-it	2.9%	18.7% (6.5×)	—
2	Qwen 3.5-4B-Base	0.0%	**75.8%**	**0.0%**
2	AfriqueQwen-8B (20-African-lang pretrained)	0.0%	16.5%	—

The round-2 control is the decisive experiment: same model, same token budget, same training procedure — targets shuffled across records. It scored **0.0%** while the real data scored **75.8%**. Notably, the control model *did* learn the surface format (its outputs look like Bantu numeral phrases) and still got nothing right: **format-learning contributes zero accuracy; the entire lift is the content of the data.**

Round 1's fine-tuned model achieved a perfect 24/24 on held-out arithmetic — long compositional numeral phrases that cannot be memorized from 1,027 training examples — and round 2 reproduced this at scale (183/210). The models are learning the *system*, not the examples.

Total cost of the entire program, including every failed run: about **$41 of GPU time**.

1. Objective and thesis

BantuNomics produces machine-verified linguistic data for Bantu languages. The commercial thesis requires demonstrating, credibly, that this data improves AI models on tasks they currently fail. "Credibly" imposes four requirements this program was designed around:

Held-out evaluation — no test item (or near variant) in training.

Exact-match grading — the harshest regime; no partial credit.

A control condition — proof the lift is the data's content, not fine-tuning per se.

Full receipts — frozen manifests, sample predictions, honest incident reporting.

2. Data methodology

2.1 Generation through the live application

All records are generated **through the production Bantu Calculator engine** — the same code that serves bantucalculator.com — never a re-implementation (`scripts/common/app_bridge.py`). Every target is therefore machine-verified against native-attested cartridge data at generation time. Round 2 generated against the live 10-language engine (isiZulu, isiXhosa, siSwati, Kiswahili, Sepedi, Sesotho, Silozi, iciBemba, Kinyarwanda, Chitonga — the last added to production the same week).

2.2 Record model and splits

One record = one **unique semantic item** with a stable `record_id`, provenance, accepted alternative forms, and a language-independent `split_group_id` that keeps related items (same number across directions/templates) in the same partition. Splits are **grouped** (train/validation/test ≈ 79/10/10) with a **zero group-leakage invariant** and an **immutable test manifest** (SHA-256).

	Round 1	**Round 2**
Unique semantic records	1,273	26,517
Validation pass rate	100%	100%
Train records	1,027	20,999
Frozen test set	139	2,773 (evals on a seeded 400-case stratified sample)
Group leakage	0	0
Contamination result	CLEAN	CLEAN (9 cross-group surface collisions detected and removed from train; test untouched)

Capabilities: numerals (both directions), arithmetic (worked expressions), calendar (names + full dates), noun-class counting (concord), tutor items. Round 2 rebalanced coverage: calendar ×2.2, tutor ×7.5 versus round 1.

2.3 The control dataset

`control_shuffled_corrupted`: the exact training records with **assistant targets shuffled across records**. Inputs, languages, token budget, and format are identical; the input→target mapping is destroyed. Any accuracy a model gains from this data measures format-adaptation, not knowledge. Round 2 used the control twin of the same stratified subset as the real run (5,840 records each).

2.4 Evaluation

`evaluate_text_model.py`: greedy decoding, deterministic **accepted-form scoring** — a prediction scores only if it exactly matches a native-attested target or a recorded alternative (variant-tolerant where translators confirmed variants; nothing else). Per-capability breakdowns and five sample predictions are saved into every eval JSON for human sanity-checking. Known decode traps are neutralized (see L1).

3. Experimental design

Element	Round 1	Round 2
Models	Qwen3.5-4B-Base; Gemma-4-E4B-it	Qwen3.5-4B-Base
Adaptation	QLoRA (NF4), r=16, α=32; Qwen: `target_modules=all-linear`	same
Schedule	3 epochs on 1,027 (≈195 steps, eff. batch 16, seq 512)	2 epochs on 5,840 stratified (=730 steps)
Conditions	baseline vs fine-tuned	baseline vs fine-tuned vs **control**
Test set	139 held-out	400-case stratified sample of the frozen 2,773

The round-2 subset kept **all** calendar (290) and tutor (550) records and capped arithmetic (2,000), numerals (1,500), noun-class (1,500) — coverage where it was scarce, efficiency where it was abundant.

4. Infrastructure

Modal A100-40GB, function-per-run (`modal_runner.py`); image = debian-slim + pinned-

by-smoke-test transformers/peft/trl/bitsandbytes stack; scripts/configs/data shipped into the image; adapters/evals persisted to volume `bantunomics-finetune-runs`; base weights cached on `bantunomics-model-cache`; HF datasets cache **container-local** (L9).

Smoke-first discipline: every model/config runs a 5-step smoke (~$0.25) before any

full run; smoke failures caught 4 of the 11 lessons before they cost anything.

Gated-model access via a workspace `huggingface-secret`; all credentials handled

value-blind end to end.

5. Results

5.1 Round 1 (139 held-out cases, exact match)

Capability	Gemma base	Gemma FT	Qwen base	Qwen FT
numerals	4/64	23/64	5/64	**39/64**
arithmetic	0/24	0/24	0/24	**24/24**
noun_class	0/22	1/22	0/22	**12/22**
calendar	0/16	2/16	0/16	0/16
tutor	0/13	0/13	0/13	4/13
Overall	2.9%	18.7%	3.6%	**56.8%**

Findings: (a) stock models produce *fluent, parsable, wrong* answers (invalid-rate 0) — the fluent-but-wrong failure mode, quantified; (b) the same data lifted Qwen 3× more than Gemma — **model choice dominates**; (c) perfect held-out arithmetic indicates rule acquisition, not memorization.

5.2 Round 2 (400 frozen held-out cases, exact match)

Capability	Baseline	Control	Fine-tuned
numerals	0/74	0/74	**70/74 (95%)**
arithmetic	0/210	0/210	**183/210 (87%)**
noun_class	0/46	0/46	24/46 (52%)
calendar	0/30	0/30	4/30 (13%)
tutor	0/40	0/40	22/40 (55%)
Overall	**0.0%**	**0.0%**	**75.8%**

The baseline zero was verified legitimate by inspecting sample predictions (English markdown meta-answers; the model does not attempt Bantu). The control's zero was verified equally carefully: its outputs are Bantu-formatted numeral phrases — format learned, content absent.

5.3 AfriqueQwen-8B: the value-add-over-African-pretraining comparison

McGill-NLP's AfriqueQwen-8B — pretrained on 20 African languages — was run under the identical lean protocol (same data, same 730 steps, same frozen test sample):

Capability	Baseline	Fine-tuned
numerals	0/74	26/74
arithmetic	0/210	20/210
noun_class	0/46	11/46
calendar	0/30	**9/30 — the best calendar score of any model**
tutor	0/40	0/40
Overall	**0.0%**	**16.5%**

Two findings: (a) **African-language pretraining contributed nothing at baseline** — 0.0%, answering in math-homework boilerplate without attempting Bantu; (b) it also did not make the model a better student of this data: 16.5% versus the 4B Qwen3.5's 75.8% under the identical protocol — L2 (model choice dominates) at full force. Caveat: one run, no per-model hyperparameter search; the comparison is same-protocol, not best-effort-per-model. Its calendar strength suggests complementary pretraining knowledge worth a targeted follow-up.

5.4 Combined findings

The data works, and the control proves why: verified targets 75.8%, shuffled

targets 0.0%, all else equal.

Coverage is a dial: rebalancing moved overall accuracy 56.8% → 75.8% on a *harder*

test set; calendar moved off zero exactly in proportion to its (still small) share. Remaining weak spots map to remaining coverage, not to a model ceiling.

Compositional generalization is real: 87–100% on held-out arithmetic across

rounds — phrases like *icyenda kuba cumi na karindwi bingana na ijana mirongo itanu na gatatu* (9×17=153, Kinyarwanda) generated character-perfect.

Exact-match understates knowledge: many numerals misses are one-concord-off;

measured accuracy will also drift upward as translators confirm more accepted variants (the keep-alternatives pipeline feeds the eval directly).

6. Engineering lessons (L1–L11)

Paid-for knowledge; each generalizes beyond this program.

#	Lesson	Consequence
L1	Qwen-family chat templates default to thinking mode; `<think>` consumed the whole generation budget and a well-trained model scored 0/139	Evals must pass `enable_thinking=False` + strip think blocks; every eval saves sample predictions for sanity
L2	Same data, Qwen 3× Gemma	Report data value per-recipient-model; pick the vehicle deliberately
L3	Coverage holes become zeros verbatim	Coverage audits before training; stratified subsets keep scarce capabilities complete
L4	Without a control you can't attribute the lift	Shuffled-target twin, same token budget — cheapest credibility available (~$5)
L5	Version drift breaks something every run (TRL `max_seq_length`→`max_length`; `apply_chat_template` returning `BatchEncoding`; PEFT unable to auto-map new architectures)	Adaptive shims in-repo; smoke-first always
L6	The data generator silently pointed at a stale (moved) copy of the application	Generation bridge pins the live repo; data now provably reflects shipped engine
L7	Real cost ≈ $1–2 per small run	Iterate freely; controls and multi-seed are affordable
L8	Results in terminal output don't persuade anyone	`build_dashboard.py` → self-contained comparison UI, regenerated per eval
L9	A shared HF cache volume served a **stale dataset** at an unchanged path — a "20k run" trained 65 steps on old 1k data	`HF_DATASETS_CACHE` container-local; **always verify `max_steps ≈ len(data)×epochs/batch`** before trusting a run
L10	Git Bash rewrites `/pkg/...` container paths in CLI args	`MSYS_NO_PATHCONV=1` on every run with path args
L12	A **stale scripts image-layer** served an old eval script (think-strip present locally, absent in-container) — diagnosed byte-level after a 0.0% that hid character-perfect answers under think-tag debris	Robust regex think-strip (leading/trailing/partial tags); **`eval_script_version` stamped into every eval JSON** so staleness is self-evident; env cache-buster forces fresh layers
L11	Two 20,999-row trainings hit the 4h function timeout with nothing persisted (~8 GPU-h lost): all-linear LoRA on an MoE ≈ 9–10 s/step	Throughput-budget (steps × measured s/step + 30% margin) before launch; checkpoint by steps; volume-commit in a `finally` block

7. Cost accounting (honest)

Item	Approx. cost
Round 1 complete (2 trains, 6 evals, probes, smokes)	~$7
Round 2 successful work (baseline, lean pair, 2 evals, smokes)	~$12
Round 2 failures (stale-cache run + eval; 2×4h timeouts)	~$20–22
Total	**≈ $41**

Even including every failure, the marginal cost of the *complete scientific result* — baseline, control, and a 75.8% fine-tune — was on the order of a restaurant dinner.

8. Threats to validity and limitations

Single seed per condition. The shared config specifies a 3-seed protocol (42/1337/2718) for mean±CI; not yet run (~$10). The effect sizes (0→75.8 vs 0→0) dwarf plausible seed variance, but the CI pass is the remaining rigor step.

Round-2 eval used a 400-case stratified sample (seeded, documented) of the frozen 2,773-case test set, for eval-cost reasons; final scoring on the full set is pending.

Rounds are not comparable to each other (different test sets/difficulty); all comparisons are within-round. The dashboard enforces separate axes.

Accepted-form completeness: some misses may be legitimate variants not yet recorded; this biases *against* the fine-tuned model (conservative direction).

Benchmark scope: scores measure engine self-consistency with native-attested data, not independent human adjudication (that axis is designed but not yet run).

One model family carried round 2; Gemma serves as the round-1 sensitivity contrast.

9. What these results license us to claim

✅ "Fine-tuning on BantuNomics data took a stock 4B model from 0% to 76% exact-match on held-out Bantu numeracy for ~$5 of compute."

✅ "A matched control (same tokens, shuffled targets) scored 0% — the entire lift is the content of the data."

✅ "Held-out compositional arithmetic reached 87–100% — system learning, not memorization."

✅ "Capability gaps track data coverage, not model limits — coverage is a dial we control."

✅ "A model pretrained on 20 African languages scored 0% before our data and 16.5% after — African-language exposure alone neither solves Bantu numeracy nor substitutes for verified structured data."

⚠️ Not yet claimable: seed-averaged CIs; full-test-set scores.

10. Reproducibility

```
# data (local, CPU)python scripts/generate_dataset.py --config
configs/data_generation/round2_scale.jsonpython scripts/validate_dataset.py && python
scripts/create_splits.pypython scripts/contamination_checks.py          # must end CLEANpython
scripts/export_dataset.py --split train --fmt instructionpython scripts/export_dataset.py --split test
--fmt eval# training + eval (Modal; smoke first, always)MSYS_NO_PATHCONV=1 modal run modal_runner.py -
-model qwen35_4b_r2 --smoke --data /pkg/data/normalized/train.lean6k.jsonlMSYS_NO_PATHCONV=1 modal run
modal_runner.py --model qwen35_4b_r2 --tag <tag> --data <train-or-control.jsonl>MSYS_NO_PATHCONV=1
modal run modal_runner.py --model qwen35_4b_r2 --evaluate-run <tag> --cases
/pkg/data/normalized/test.eval.sample400.jsonl# report UIpython scripts/build_dashboard.py
# -> results/dashboard.html
```

Artifacts: eval JSONs in `results/evals/` (invalid runs preserved under `_invalid/`), adapters + checkpoints on the Modal volume, manifests in `data/manifests/`, this report and the dashboard in `results/`.

11. Future work (ordered by evidential value per dollar)

3-seed principal result with mean±CI (~$10).

Full 2,773-case final scoring of the round-2 winner (~$4).

AfriqueQwen-8B: value-add over 20-African-language pretraining (~$6).

Calendar/tutor deep-fill: second coverage-dial turn where zeros persist.

Engine-as-verifier loops (rejection sampling / RL): the deterministic calculator

grades any output at zero labeling cost — the designed path from 76% toward saturation, and a capability few language domains possess.

Syllable-aware tokenization study: exact-form generation is the regime where BPE's

misaligned segmentation taxes hardest; the syllable-floor hypothesis is directly testable on this benchmark.

Prepared 2026-08-08. Data, adapters, and full run history available under the terms in `LICENSE_AND_PROVENANCE.md`. All results reproducible from the commands above.