# Poro-2-8B in production: what we measured, what broke, what we built around it

This is a working reference for how Poro-2-8B behaves in two of my projects, and the deterministic machinery added to make it usable. The promise in the title is that every number here traces to a run you can repeat, or to a judgment I name as one. Where a claim rests on a small sample or a single rating, I say so in the same sentence.

Two projects measured Poro independently and reached opposite deployment decisions from the same evidence.

- **feedback-intelligence** (this repo) structures retail feedback and writes a Finnish management summary. It adopted Poro for both jobs. The categorization findings below are from here.
- **mikkonumminen.dev** is a portfolio chat backend (Postgres, pgvector, Ollama). It ran a blinded, statistically tested comparison of Poro against qwen3:8b and llama3.1:8b, then kept its resident qwen2.5:7b for production and used Poro only to map the Finnish-quality ceiling. The language-drift findings below are from there.

The same 26/30 naturalness result led one project to adopt Poro and the other to pass on it. The two decisions are not in conflict, because the right call depends on how the output is consumed. The last section works that out into a rule.

## How to reproduce these numbers

The categorization probes below were run on one machine: an RTX 3080 Ti (12 GB), Ollama 0.30.10, model `hf.co/mradermacher/Llama-Poro-2-8B-Instruct-GGUF:Q4_K_M` (a 4.9 GB Q4_K_M quantization), on 2026-07-15. All LLM calls run at temperature 0 and are prompt-byte-stable, so a given input yields a given output (ADR-0018).

To repeat the categorization probes: rename `domains/retail/category-keywords.json` aside to disable the deterministic override, restart the desk API, and POST texts to `/interpret`; the returned `structure.category` is Poro's raw pick. The structuring quality harness is `tools/FeedbackIntelligence.StructuringEval`. The naturalness ranking and the Voikko rates below come from the mikkonumminen.dev eval set (`chat-backend/evals/`), not from this repo, and were measured on the same hardware.

Not everything here reproduces the same way. The category probes and the Voikko rates can be reproduced by re-running code, because the model is deterministic at temperature 0 and the answers are saved. The naturalness ranking cannot: reproducing it means finding a second judge, not re-running a job.

The model is cheap to serve: about 5.9 GB of VRAM loaded, an 8192-token context window, and the largest prompt-plus-output we measured was 5205 tokens.

## What Poro is good at: Finnish

On a blind ranking of 30 Finnish answers, Poro placed first in 26 of 30 rounds, mean rank 1.37 against qwen3's 2.23 and llama's 2.40 (Friedman chi-square 22.85, p below 0.0001; pairwise Poro over qwen3 20 to 3, over llama 22 to 1). It was never sole-worst; qwen3 was worst 9 times and llama 11.

One caveat the number does not carry on its own. The ranking was made by a single native speaker, blinded to which model produced which answer, who is also the author of the project that went on to adopt Poro. The Friedman test says the ranking was internally consistent across questions. It does not say the judge was disinterested. So 1.37 means one blinded native speaker, who has a stake, preferred it strongly and consistently. A second judge with no stake would be the obvious next check.

Poro's edge is morphological. It inflects technical terms the way a Finnish speaker does ("Astro 6:sta", "TypeScriptistä") where the general models produce stiff or half-anglicized forms.

A deterministic spelling and grammar pass (Voikko, all three models, 84 scored answers each) put Poro at 3.3% flagged tokens, qwen3 at 4.0%, llama at 5.8%. The 84 is larger than the ranking's 30 because the blinded human ranking took a subset a person could get through by hand, while the automatic Voikko pass scored the full set. Taken flat against a 7.2% floor measured on human-approved UI strings, 3.3% would read as "Poro writes more correctly than a person," which nobody should claim. The likelier reading is that Voikko flags proper nouns, code identifiers and anglicisms, and human text carries more of those. There is a tension worth naming: I also claim Poro inflects technical terms better, which means it produces more of exactly the tokens Voikko tends to flag. If it produces more of them and still earns a lower flag rate, the result is stronger than it looks. If it produces fewer, the low rate just means it writes blander Finnish. I have not separated the two, and one controlled pass (same token classes, counted) would settle it. Until then, 3.3% is suggestive, not decisive.

On grounded synthesis Poro is level with the best of the three, not ahead: substantive grounded Finnish 25 of 27, tied with qwen3, well above llama's 18. The naturalness edge does not cost synthesis quality. It costs something else, which the next two sections describe.

## What Poro is bad at: structured decisions

### Categorization is inconsistent, not uniformly wrong

My first instinct was to write "Poro has a strong attractor toward the dairy category." A controlled probe says that is too strong. I fed ten items that contain no "maito" substring and read the raw category with the override off:

Each row is the exact text POSTed to `/interpret`.

| Text fed (no "maito" substring) | Poro's raw category |
|---|---|
| `Ostamani banaani oli täysin mustunut.` | hevi (correct) |
| `Porkkanat olivat pehmeitä ja nahistuneita.` | hevi (correct) |
| `Omenat olivat pehmeitä ja jauhoisia.` | hevi (correct) |
| `Kurkut olivat limaisia pinnalta.` | hevi (correct) |
| `Appelsiinit olivat kuivia sisältä.` | hevi (correct) |
| `Leipä oli aivan homeista.` | leipa (correct) |
| `Nakit olivat pilaantuneet jo ostopäivänä.` | maito_kylma (wrong, is meat) |
| `Mehu oli hapanta ja väljähtänyttä.` | maito_kylma (wrong, is drinks) |
| `Ostamani jauheliha haisi pahalta.` | kuiva_elintarvike (wrong, is meat) |
| `Keksit olivat vanhentuneita.` | kuiva_elintarvike (defensible) |

Six of ten correct, two to dairy, two to dry goods. The dairy pull is real but weak (2 of 10), and it is not the only wrong basin. The sharper finding is inconsistency. In an earlier run the shorter `banaani oli mustunut` landed in dairy, while the fuller `Ostamani banaani oli täysin mustunut` in the table above lands in hevi. Same model, same temperature, a different surface form, a different answer.

Two mechanisms are worth keeping apart, because I bundled them in the first draft. The first is substring capture: "maitosuklaa" (milk chocolate) reliably goes to dairy because "maito" sits inside the word, and this is deterministic, always wrong, and a wordlist fixes it cleanly. The second is a weak, phrasing-sensitive prior: bare product words sometimes drift to dairy or dry goods (nakit, mehu, jauheliha above), unreliably. A wordlist that forces the right category also makes an inconsistent model consistent, and that consistency is worth more here than any single correction.

## A few things it will not do at all

- It emitted an optional model-authored `sentiment` field zero times out of 71 on an announced seed-42 run, despite the field being in the schema (ADR-0031). We stopped asking and derive sentiment deterministically instead.
- It shows no sign of self-identifying garbage. This is a probe, not a settled result. Three obvious nonsense items were routed to a dedicated `ei_palautetta` bucket zero times, and we dropped the bucket (ADR-0032). Three items is too few to conclude "cannot," the same small-sample problem as the four-item containment claim below. Read it as no signal on three tries, pending more items.
- Its JSON discipline on messy Finnish is unmeasured and, from the failures we have seen, unreliable, which is why the salvage layer is mandatory rather than best-effort (ADR-0004).

## Language drift (from mikkonumminen.dev)

The portfolio project measured a different family of failures, all around language and instruction-following.

- Mid-answer language drift: a Finnish question answered in English or the reverse. Poro, tuned Finnish-first, tends to answer in Finnish even when the question is English.
- Translation overstep: it translates meaning-bearing proper nouns however firmly the prompt forbids it. "kasvulabs" came back as "Growth Labs" in a live run, because "kasvu" is Finnish for growth.
- Appended commentary: after a correct first-line translation it adds an unbidden "However, considering...".
- Code-dense Finnish confuses the routing: identifier-heavy Finnish dilutes the a-and-o language heuristic, so such a question can be answered in English.
- Containment could not be ranked at that sample size. An earlier "Poro is worst at containment" line was retracted after a rate limiter contaminated the run.

## What we built around it

The rule both projects converged on is that the model lives in exactly two places, structuring messy input and writing prose, and a deterministic layer runs in front of it and outranks it (ADR-0006). Everything here is that layer.

In this repo:

- An alert lexicon forces safety, payment, legal and racism categories from a wordlist before the model runs, and the model cannot remove a hit (the safety/payment/legal set is established in ADR-0006; racism as a forced category in ADR-0027).
- A category-keyword override forces product departments and, as a fallback, service and premises, with cross-category exclusions so compounds route correctly (ADR-0036, ADR-0037). A corpus false-positive scan checks the wordlists before they ship, and it is worth more than the override itself. It caught "kana" (chicken) matching the common word "aikana", "kala" matching the fish-department name in a comment about the fish counter, and bare "kassa" pulling cleanliness complaints into checkout service. Each would have shipped as a wrong category.
- A mandatory salvage layer catches malformed JSON and preserves the raw text on failure (ADR-0004).
- Hints nudge the model on confusable boundaries and on names the wordlist has not enumerated (ADR-0028, ADR-0035).
- Sentiment is derived from the type the model already assigns, with no second call (ADR-0030).
- Grounding is structural: a synthesis claim whose cited feedback IDs fail validation is dropped rather than shown (ADR-0009).
- A desk correction loop lets a human fix what the rules miss, and measures the model-versus-human correction rate so we know which departments to harden next (ADR-0035).

In mikkonumminen.dev, the same shape against the drift failures: a known-entity map restores proper nouns Poro translated away (the direct analogue of our wordlist), moving the Finnish-path language anchors from one to three lifted adherence on code-heavy Finnish from 85% to 100%, translation output is truncated to the first line to stop appended commentary, and keyword task-gates block off-task generation before it reaches the model.

The naturalness edge is not free. It is paid for with the layer above: a salvage pass, several wordlists, a validator, a correction loop. The price buys a second thing beyond making Poro usable: the same layer would protect whatever model comes next, including a better one, because it constrains the decision rather than the model. The same wordlist would correct a different model's category mistakes without any change to it.

## When to adopt Poro, and when not to

The two projects disagree, and the disagreement is the reusable part. mikkonumminen.dev kept qwen, which makes it evidence against a naive "always take the best writer" reading. I have not run the honest control, which would be patching qwen3's faults with the same deterministic layer and comparing, so I cannot claim Poro-plus-layer beats qwen-plus-layer. What the two outcomes do imply is a rule.

Adopt Poro when the output is read by a human and the structure is enforced outside the model. feedback-intelligence fits: a person reads the Finnish summary, and categories, grounding and sentiment are all forced by the deterministic layer, so Poro only has to write well.

Keep your resident model when the model itself has to obey format and scope in-band, with no external gate. The RAG chat fits: the answer goes straight to a user, language and scope have to hold inside the generation, and Poro's drift is exactly the thing you cannot gate after the fact.

Two projects is a small sample. What makes the split between them useful is that it came from the same evidence: the deployment decision depended on how the output is consumed, not only on which model writes the best Finnish.