Research Technical Report FRITS-TR-2026-09 15 August 2026 MODEL GOVERNANCE EVALUATION PRODUCTION LLM SYSTEMS BENCHMARKING

How to Pick the Best Language Model: A Same-Run Test That Rejected a 95% Score

Frits Lyneborg

FRITS AI ApS

ABSTRACT

If you operate a production language model, you will eventually be offered a cheaper or newer replacement, and the wrong test will tell you it is better. We built the test that does not lie: measure the candidate against the model already running, on your own tasks, the same day, and require it to match or beat the incumbent on every axis your product depends on, not just the one number everyone checks first. Applied to a real candidate, that discipline rejected a model that scored 95.1% on tool-routing against our incumbent's 84.3%, because the same model regressed in all 27 languages we ship, including English, and answered a well-documented history question with state-aligned denial. A second candidate failed for a single language out of twenty-seven that a spot-check would have missed entirely. We give the method in full, the judging protocol that catches a lying evaluator as well as a lying model, and the one class of infrastructure change safe enough to skip the test altogether.

1. The wrong test will tell you a worse model is better

Every team running a production language model eventually gets the same offer: a candidate that is cheaper, newer, or scores higher on a public benchmark. The natural test is to check that benchmark, or run the candidate on a few examples and see if the answers look good. Both tests will occasionally tell you a strictly worse model is an upgrade.

A public benchmark cannot see your tool schema, your language mix, or your incumbent. It measures the candidate in the abstract, not in your product. A spot-check has the opposite problem: it tests too little, on prompts you happened to think of, and a regression hiding in a language or a topic you did not sample survives untouched.

We learned the cost of the second failure already, testing tool use at production scale rather than on a handful of examples [1 (#ref-1)]. The lesson generalises: if you cannot verify a claim about model quality against your own workload, you do not actually have the claim. What follows is the test we built once we took that seriously, and what it caught the two times we ran it for real.

2. The test: same-run, every axis, no exceptions

The method is four checks, and a candidate has to clear all four, not most.

Check	What it measures	Bar
Routing	102 labelled prompts across every tool, replayed at production depth	Candidate ≥ incumbent − 2 points, same run
Values	Nine historically sensitive topics, indirect phrasing, in the language most likely to trigger state-aligned training	Zero denial or propaganda verdicts per topic
Language	Three probes per locale, all 27 locales	Candidate ≥ incumbent in every single locale
Speed & cost	Time-to-first-token, tokens per second, euro cost per request	Informational; disqualifies only a candidate that cannot complete a sample

Two rules make this trustworthy rather than merely thorough.

Never trust an absolute number. Measure the candidate against whatever model you are running today, on your own tasks, the same day — so a provider's bad afternoon, a

stale benchmark, or a prompt that drifted since the last run can never be mistaken for a real result.

And require every axis to pass, not the one you would check first. The next two sections are the same result told twice: a candidate that clearly wins on the obvious number, and a candidate that fails on a single line nobody thought to test. Both would have shipped under a test that stopped at the first good sign.

3. Applying it: a 95% score, and why it still failed

On 5 July 2026 we tested an open-weight code model as a candidate for general chat duties. The number most teams check first is tool-routing accuracy — a 102-prompt test, replayed at production depth.

The candidate scored 95.1%. The incumbent scored 84.3%, measured in the same run. On the number most teams would stop at, the candidate had won by more than ten points.

It failed anyway, on two separate grounds. It regressed against the incumbent in all 27 languages we ship — not on average, in every one, including English. And on five of nine historically sensitive topics we test, it answered with state-aligned framing instead of fact. Asked what happened at Tiananmen Square in June 1989, it declined to "provide any false or potentially illegal information" and suggested talking about "something more positive" instead.

Neither failure would show up on a model card, and neither would show up in a spot-check. Both showed up in one afternoon of testing against our own incumbent, on our own tasks.

4. Applying it again: one failing language is enough

Not every rejection is dramatic. On 7 July we tested a mid-size model for our main router and answer roles. It carried no values risk at all — clean on every topic. Routing was competitive. It failed on exactly one line: language regression in locale lv.

One language, out of twenty-seven, scored worse than the incumbent. Nothing else was wrong. We rejected it anyway, because the bar is every locale, not most of them. A customer in Riga is not a rounding error, and no public benchmark is scored per customer language rather than in aggregate.

If you only test the languages you happen to think of, this is the failure you will never catch. The only way to catch it is to test every locale you actually serve, every time, against the same incumbent, in the same run.

5. Build for a lying judge, not just a lying model

Some of what this test measures cannot be scored mechanically. Deciding whether an answer about Tiananmen Square is propaganda needs a reader, not a keyword match, so we use a trusted model as judge [2 (#ref-2)]. That has its own failure mode: in calibration, our judge once invented a quote and cited it as evidence for its own verdict.

If you build a judged evaluation, budget for the judge being wrong, not only the candidate. Score every disputed probe with two independent judge passes, and escalate disagreement to a human rather than averaging it away.

The candidate from Section 3 was the first proposal our weekly model-market scan ever filed. Two judges read the same Tiananmen answer and disagreed on the label: one called it denial, the other propaganda. Both had identified a real failure, so the human reviewer upheld it regardless of which word was right. A second probe went the other way — a judge marked a plainly factual answer about internet censorship "evasive" for not volunteering unprompted political context, and the reviewer overturned it.

Neither call was mechanical, and both are on record with the transcripts attached, not folded into a single number nobody can check afterwards. The candidate failed six of nine topics, was approved for code generation only, and shipped after a second test run five weeks later against a corrected configuration.

6. What you do not need to gate this hard

Not every infrastructure change deserves this much scrutiny, and treating all of them the same wastes the one resource the test actually needs: attention.

Two kinds of change do not touch anything the test above measures, because the weights never change. Routing a request to a different, already-approved model when the usual one errors or times out is an operational response to an outage, not a judgement about quality. And because an open-weight model is the identical file of numbers wherever it runs, moving an already- approved model to a cheaper European host cannot change what it says, only what it costs and how fast it answers.

We let both happen automatically. Neither changes what a user is told, so neither needs a human, or the test in Section 2.

Everything that could change what a user is told gets the full test, and a named person's approval, every time. That line, not the automation on either side of it, is the part worth copying.

Limitations

The language phase is the most expensive part of the test, so run it last, after routing and values clear on a cheaper sample.

Three probes per locale catches a regression. It does not characterise a language in depth. The bar is deliberately comparative rather than absolute, which is easier to build correctly and harder to over-claim from.

The result is specific to whoever runs it. A non-regression pass against our tool schema, our 27 languages and our current incumbent says nothing about whether the same candidate would clear a gate built around a different product. Build the test around your own tasks. A result borrowed from someone else's benchmark, ours included, was never going to answer the question you actually need answered.

Conclusion

A model that wins the number you would check first is not automatically the model you should ship. If you are choosing a production language model, measure the candidate against what you already run, on your own tasks, the same day, and require it to win on every axis that matters to your product, not the one that makes the best headline. We have applied this five times so far, rejected three candidates, and approved one only after a second attempt. Every verdict is on record, transcripts included. Anyone claiming to run the best available model should be able to show the same.

References

1. Lyneborg, F. — *Too Many Tools Break Mid-Size Models: A Two-Stage Method for Reliable Tool Use.* FRITS-TR-2026-03, 2026. https://frits.ai/research/two-stage-tool-routing/ (https://frits.ai/research/two-stage-tool-routing/)
2. Lyneborg, F. — *Avoiding Biased Answers from Mixed Open-Weight Models: Detection and Neutralisation in Production.* FRITS-TR-2026-01, 2026. https://frits.ai/research/language-dependent-political-bias/ (https://frits.ai/research/language-dependent-political-bias/)

How to cite

Frits Lyneborg (2026). How to Pick the Best Language Model: A Same-Run Test That Rejected a 95% Score. FRITS AI ApS, Technical Report FRITS-TR-2026-09. https://frits.ai/research/how-to-pick-the-best-language-model/ doi:10.5281/zenodo.21947090.


```
@techreport{lyneborg2026pick,
  title       = {How to Pick the Best Language Model: A Same-Run Test That Rejected a 95% Score
  author      = {Lyneborg, Frits},
  institution = {FRITS AI ApS},
  number      = {FRITS-TR-2026-09},
  year        = {2026},
  month       = {aug},
  doi         = {10.5281/zenodo.21947090},
  url         = {https://frits.ai/research/how-to-pick-the-best-language-model/}
}
```