

FertiScope: Measuring the Multilingual Tokenizer Tax in Low-Resource Asian Languages

Leo Dang, Antoine Pedretti, Miles Whitticker, Vinh Van — with Apart Research at the *Global South AI Safety Hackathon, Da Nang, June 2026*

Abstract

When using LLMs with lower-resource Asian languages, a hidden “tax” is applied where text is fragmented into a greater number of tokens compared to the English language. That greater token generation raises API bills, leaves room for fewer in-context examples, and fills the context window faster. To address this we made FertiScope: an open-source web tool that measures how many tokens are generated in 15 lower-resource Asian languages as well as other associated costs. Our tool measures three tokenizers: GPT-4o, GPT-4/3.5 and SEA-LION v3. We find that on parallel FLORES-200 sentences, the cost of the same content reaches up to 11.6× English, depending heavily on the tokenizer. Finally, we carried out a 540 call needle-in-haystack test across a subset of language, model and tokenizer combinations which found no degradation at token length inside the models’ publicly advertised context window sizes.

1. Introduction

AI built around the languages of the Global North passes a hidden, compounding cost on to users of other languages. Through one technical choice, the tokenizer, users in lower-resource-language regions can pay several times more for a measurably worse service, and the gap grows with conversation length.

A tokenizer’s *fertility* is the number of tokens it produces per word. English sits near 1.3 tokens per word (1.26 on our FLORES corpus). Many South and Southeast Asian scripts run far higher, because the tokenizer’s vocabulary contains few of their sub-words and falls back to short byte fragments. Since every commercial API bills per token and every context window is finite in tokens, high fertility carries three linked costs, first catalogued by Petrov et al. (2023) and Ahia et al. as a “double penalty”: higher cost for the same content, fewer in-context examples per prompt, and a smaller usable context budget that fills faster across turns.

This is a safety problem as well as an economic one. Failure in lower-resource languages is often qualitatively different from English failure, not just more frequent. Hallucinations are more often a total collapse than a plausible partial error: in the Mu-SHROOM shared task (SemEval-2025 Task 3), the majority of Tamil hallucinations fall in the most severe category. Yong et al. 2023 found that safety alignment transfers poorly across languages with an unsafe prompt translated into a low-resource language raising attack success against GPT-4 from under 1% to about 79%. This high one-off

failure rate is hidden under an otherwise acceptable average safety score. Sayeedi et al. (2025) find that capability gaps are large and systematic across language families and domains. Among these failure modes, the tokenizer cost is the one that is fully measurable in advance, computable locally, and, as our results show, often repaired by a procurement choice rather than retraining. Algorithmic bias across languages and dialects in multilingual and open-source models is of concern to the Global South, so it's relevant to have a measurement that a regional team can run on its own text before it ships.

We do not claim that fertility *causes* accuracy degradation; our research plan shows that claim is hard to identify, and we avoid it. We instead answer questions a deployer can act on: how large is the tokenizer cost across these languages, how much does it vary by tokenizer, what does it cost in dollars and usable context, and can a team see all of that before choosing a model?

Findings:

1. **FertiScope**, an open-source web tool that, for arbitrary text or for 15 lower-resource Asian languages, computes tokenizer cost and reports a dollar multiplier, in-context capacity, and a context-budget curve across three production tokenizers, locally and in seconds. To our knowledge it is the first such tool aimed at a deployment decision rather than research; prior art is papers, scripts, and generic token counters.
2. **A leaderboard** of same-content cost across 16 languages (English plus 15 Asian languages) and three tokenizers on parallel FLORES-200 sentences. The cost is large but tokenizer-dependent, and it varies by writing system: the Llama-3.1 tokenizer that SEA-LION uses helps the languages it already covers but not the scripts it does not.
3. **A measured/estimated split**. Every cost number is exact, however we cannot measure multi-turn accuracy risk. Instead it is shown as a bounded estimate because the research on its size is inconclusive.

2. Related Work

FertiScope builds on a well-studied problem. Our contribution is turning the research statistic into a deployment decision.

Tokenizer fairness. Petrov et al. (2023) shows that the same content can differ up to 15× in token length across tokenizers on FLORES-200, framed as a fairness statistic measured once over a reference corpus. FertiScope computes the same primitive but projects it into a dollar figure and a per-turn context-budget curve on the deployer's own text.

Fertility and accuracy. Lundin et al (2025) reports a correlational fertility-to-accuracy slope of -0.08 to -0.18 accuracy per token/word across 16 African languages and 10 LLMs (single-turn). Nayeem et al (2025) argues fertility alone is an incomplete efficiency metric and proposes STRR. We report fertility today and treat STRR as future work; FertiScope is a fast local predictor, not an accuracy claim.

Multi-turn and long-context behaviour. Laban et al. (2025), *LLMs Get Lost in Multi-Turn Conversation* (arXiv:2505.06120), report about a 39% average single-turn-to-multi-turn accuracy drop over 200k simulated conversations, which affects English too; this is why we do not assume “English stays flat.” Liu et al. (2023), *Lost in the Middle* (arXiv:2307.03172), show over-30% drops for information placed mid-context. These show that accuracy can fall as relevant information sits deeper in, or spreads across, a longer context. FertiScope shows which languages consume that context fastest, without estimating the accuracy effect.

The regional response. SEA-LION (AI Singapore, arXiv:2504.05747) continue-trains Llama-3.1 and Gemma on a Southeast-Asian corpus; SeaLLMs 3 (arXiv:2407.19672) adds anti-hallucination training; SEA-HELM (arXiv:2502.14301) is a regional evaluation suite. These adapt model weights. FertiScope measures the tokenizer cost they leave in place: SEA-LION v3 keeps the Llama-3.1 tokenizer unchanged, so any per-language differences it shows belong to that tokenizer, not to the regional training.

3. Methods

FertiScope has two parts: a measurement-and-costing engine (Sections 3.1–3.3), which is the substance of this submission, and a multi-turn recall benchmark (Section 3.4), which we ran as a pilot and will extend in future work.

3.1 Measuring tokens and words

Fertility is tokens divided by words. Tokens come from three real tokenizers run in the browser with pure-JS encoders (no native dependencies): GPT-4o (o200k_base) and GPT-4/GPT-3.5 (cl100k_base) via js-tiktoken, and Llama-3.1 / SEA-LION v3 via llama3-tokenizer-js (Llama-3.1 and the continue-trained SEA-LION v3 share this tokenizer file, so one encoder covers both).

Words are counted with Intl.Segmenter (word-like segments only). This is needed for reproducibility as for spaceless scripts (Thai, Khmer, Burmese, Lao) word boundaries are ambiguous, so any tokens-per-word value depends on the segmenter. We therefore headline a per-sentence cost ratio, which needs no word-counting, rather than per-word fertility.

The leaderboard is computed offline on 50 parallel sentences from FLORES-200 (NLLB, Meta AI; CC-BY-SA 4.0). Because every language carries the same meaning, the ratios are comparable across languages. The same engine runs live on pasted text, so a deployer can measure their own corpus instead of a benchmark proxy.

3.2 Cost, capacity, and context budget

These three metrics are calculated deterministically from the token counts. They are exact for the stated corpus, tokenizer build, prices, and segmentation convention, but they are not accuracy claims.

- **Same-content cost ratio** = tokens for the parallel sentence ÷ English tokens, on the same tokenizer. Because price-per-token is constant within an API, this token

ratio is the bill ratio for equivalent content, and it needs no word segmentation. The tool also reports per-word fertility, and a fertility-based multiplier for a user's own word-counted workload; we keep the two distinct.

- **In-context capacity** = $L(\text{window} - \text{reply reserve}) \div \text{tokens-per-example}$.
- **Context budget** = cumulative tokens across turns against the window (4K / 8K / 32K / 128K).
- The **Cost Calculator** maps a monthly workload to a bill, the dollar gap versus English, and the cheapest model for that language.

3.3 Multi-turn risk: the one estimate

The single quantity FertiScope does not present as exact is multi-turn degradation risk, shown as a 0–4 score and labelled an estimate wherever it appears. Budget pressure (how fast cumulative tokens cross 75% of the window) scores 0–2 and is the primary driver; fertility scores 0–2 but only amplifies once the budget is already under pressure, so a roomy 128K window stays low regardless of fertility. This follows a finding from our research review: the large degradation results in the literature come from constrained windows, so a short chat on a 128K model barely dents the budget and should not be flagged. The score is reported per window, and it is not a measured accuracy drop.

3.4 Multi-turn recall benchmark (pilot)

To test directly whether high-fertility languages lose information faster as context fills, we ran a needle-in-haystack pilot: for each (model, language, context-fill level, needle position), we build a haystack of FLORES-200 sentences in that language, sized in tokens using the leaderboard's measured tokens-per-sentence, insert one marker line «SECRET-KEY = CODE» at the given depth, and ask the model to return the code. The marker avoids translation ambiguity, but because it is a Latin string it may stand out in a non-Latin haystack, a caveat we return to in §4.4. The pilot covered three models (GPT-4o-mini, GPT-3.5-Turbo, Llama-3.1-8B), four languages spanning the cost range (English, Hindi, Tamil, Malayalam), three context-fill levels (50/100/150% of an 8192-token base, about 4k/8k/12k tokens), five needle positions (5–95%), and three trials each: 540 model calls. The exact endpoints, via OpenRouter, were openai/gpt-4o-mini, openai/gpt-3.5-turbo (about 16k window), and meta-llama/llama-3.1-8b-instruct (about 128k); the largest fill (about 12k tokens) sits inside every window, so no truncation occurred. The runner and the full per-call log are in the repository (research/), so the result is reproducible.

Our research plan specifies a larger 10-turn cumulative-QA protocol with controls (a within-language length control, a constrained-window arm, and off-diagonal languages such as Bengali and Somali to separate fertility from data scarcity). The plan rates that fuller causal study “strongly suggestive, not identified” without a from-scratch retraining arm, so we leave it to future work.

4. Results

4.1 The cost is large, varies by script, and depends on the tokenizer

Table 1 reports the same-content cost ratio: the tokens needed to encode the same parallel FLORES-200 sentence, divided by English, for all 16 languages across the three tokenizers. This is the quantity that governs an API bill, and it needs no word-counting. Per-word fertility is a related diagnostic that explains *why* the count is high, but because languages differ in words per sentence the two ratios diverge; we use the per-sentence ratio for cost and report fertility separately in the repository. English encodes a FLORES sentence in about 27 tokens on all three tokenizers.

Table 1 — Same-content cost ratio (tokens per parallel FLORES-200 sentence ÷ English), sorted by GPT-4/3.5 tokenizer.

Language (script)	GPT-4o (o200k)	GPT-4 / 3.5 (cl100k)	Llama-3.1 / SEA-LION (llama3)
English (Latin)	1.00×	1.00×	1.00×
Burmese (Brahmic-derived)	3.13×	11.20×	11.20×
Sinhala (Brahmic-derived)	2.76×	8.66×	8.60×
Kannada (Brahmic-derived)	2.03×	8.64×	8.64×
Lao (Brahmic-derived)	7.31×	8.55×	7.89×
Malayalam (Brahmic-derived)	1.90×	8.45×	8.44×
Khmer (Brahmic-derived)	3.19×	8.33×	8.33×
Telugu (Brahmic-derived)	1.89×	7.97×	7.97×
Tamil (Brahmic-derived)	1.99×	7.19×	7.19×
Bengali (Brahmic-derived)	1.72×	5.73×	5.69×
Hindi (Devanagari)	1.59×	4.61×	2.50×
Thai (Brahmic-derived)	1.96×	4.30×	2.23×
Vietnamese (Latin)	1.42×	2.25×	1.36×
Filipino (Latin)	1.63×	1.94×	1.92×
Malay (Latin)	1.30×	1.61×	1.60×
Indonesian (Latin)	1.24×	1.53×	1.52×

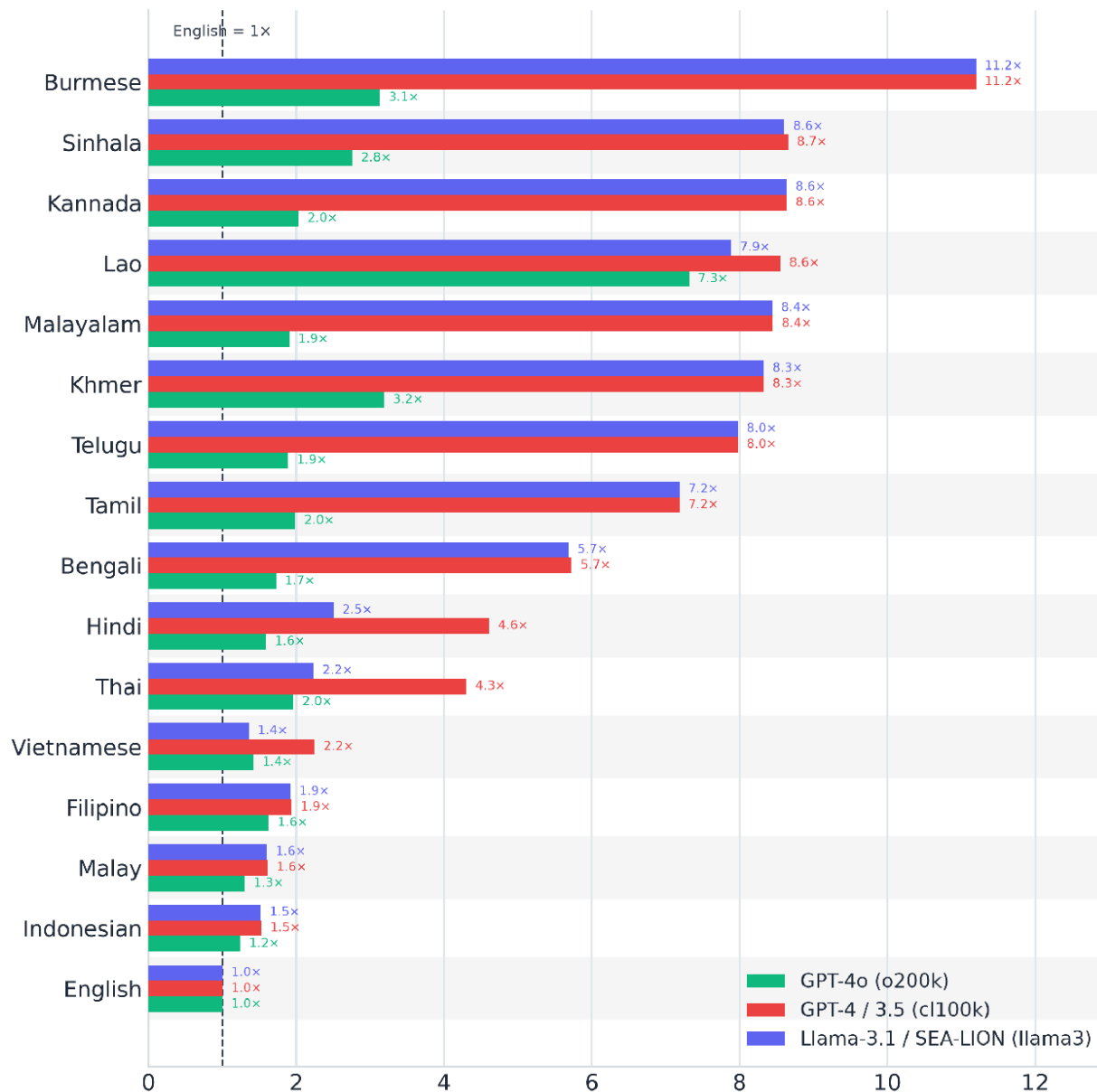


Figure 1 — Same-content cost ratio of 16 languages across 3 tokenizers.

Three observations can be drawn from the data:

- The worst-hit languages cluster by script, not by region.** On the GPT-4/3.5 tokenizer, Burmese, Sinhala, Kannada, Lao, Malayalam, Khmer, Telugu, and Tamil pay 7.2–11.2×. Thai, Hindi, and Bengali are lower (4.3–5.7×) but still well above English, while the Austronesian Latin-script languages (Indonesian, Malay, Filipino, Vietnamese) pay only 1.5–2.3×. The cost tracks how well the tokenizer’s vocabulary covers the script.
- Tokenizer choice removes most of the cost.** Switching the same content from the GPT-4/3.5 tokenizer to GPT-4o’s cuts Tamil from 7.2× to 2.0×, Malayalam from 8.5× to 1.9×, and Burmese from 11.2× to 3.1×: roughly a 70–80% reduction

in the cost ratio, from one vendor choice. Lao is the exception, still 7.3× even on GPT-4o.

- **The cost is set by the tokenizer, not by regional fine-tuning.** SEA-LION v3 keeps the Llama-3.1 tokenizer unchanged, so its per-language differences come from that tokenizer. The Llama-3.1 tokenizer encodes Hindi ($4.61\times \rightarrow 2.50\times$), Thai ($4.30\times \rightarrow 2.23\times$), and Vietnamese ($2.25\times \rightarrow 1.36\times$) more cheaply than cl100k, but it is identical, or within rounding, to cl100k for Tamil, Telugu, Kannada, Malayalam, Sinhala, Bengali, Burmese, and Khmer. A team adopting SEA-LION for a Tamil product gets adapted weights but the same token cost.

The underlying per-word fertilities match independent Indic-tokenizer benchmarks (for example, Tamil is about 12.3 tokens/word under Llama-3.1 on FLORES-200; FertiScope measures 11.25 on its 50-sentence subset). That per-word figure, about 9× English, is *larger* than Tamil's same-content cost ($7.2\times$), because Tamil packs more meaning into each word, so a per-word ratio overstates the per-sentence bill. Reporting the per-word ratio as the cost is the conflation Table 1 avoids.

4.2 Faster window exhaustion and fewer examples

Because the window is finite in tokens, fertility also limits how much conversation and how many examples fit. Both are deterministic. Figure 2 plots cumulative context use across turns, the analytical form of a window-exhaustion benchmark: on the GPT-4/3.5 tokenizer a Tamil or Malayalam conversation crosses the 4096-token window in about 3 turns, where the same English conversation lasts about 32. Figure 3 shows the effect on in-context examples: an 8192-token window holds about 60 hundred-word English examples but only 5–6 for Tamil, Telugu, Kannada, or Malayalam. The languages that most need worked examples fit the fewest.

4.3 The cost in dollars

Because price-per-token is constant within an API, the same-content token ratio is the bill ratio. On the GPT-4/3.5 tokenizer, encoding the same content costs 7.2× more in Tamil than in English: not because the Tamil request says more, but because the tokenizer emits 7.2× as many billable tokens for the same meaning. A workload that runs about \$1,460/month in English on GPT-4 Turbo (\$10/\$30 per 1M tokens, 100,000 requests/month) costs about \$10,500/month for the same content in Tamil, a roughly \$9,000/month gap that is invisible at model-selection time. FertiScope shows it alongside the cheapest alternative; switching this workload to the GPT-4o tokenizer drops Tamil's ratio to 2.0×.

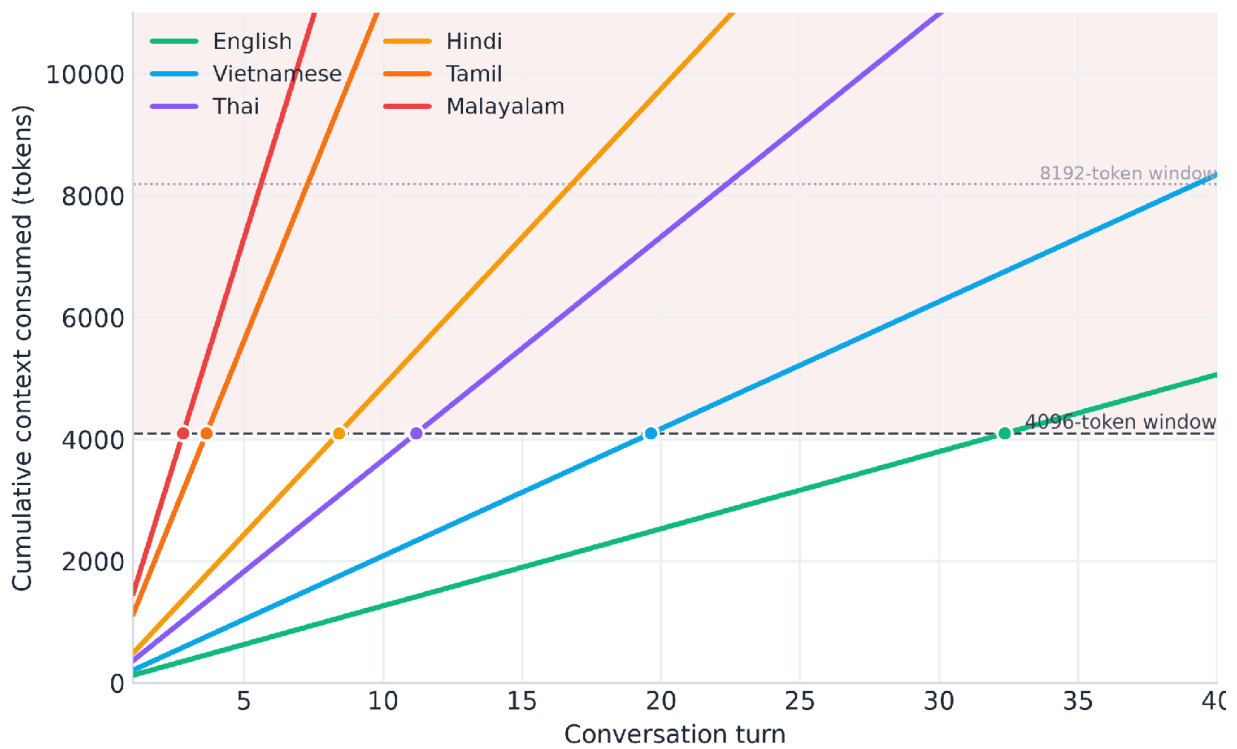


Figure 2. Context-window exhaustion by language (GPT-4/3.5 tokenizer, 100 words/turn). Dots mark where each language crosses the 4096-token window.

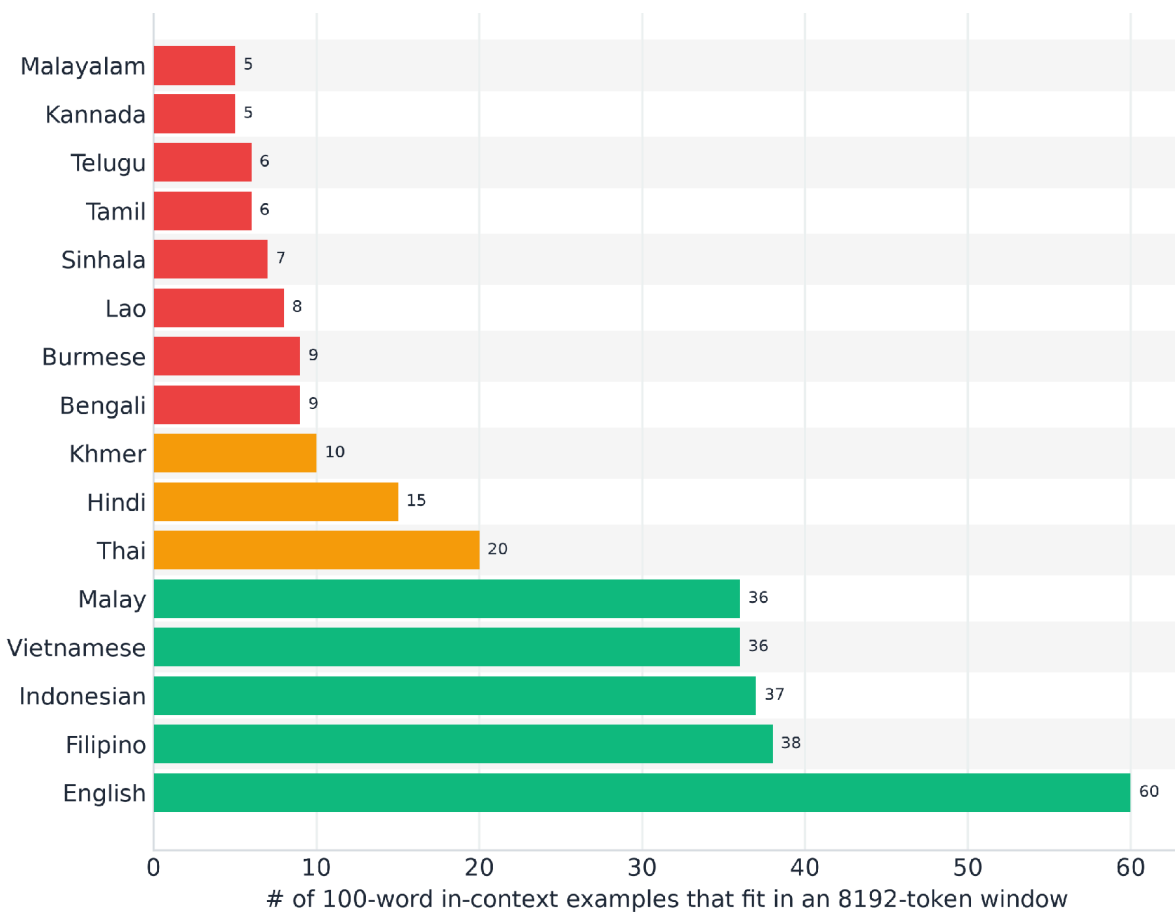


Figure 3. Number of 100-word in-context examples that fit in an 8192-token window (512-token reply reserve). High-cost languages fit roughly an order of magnitude fewer

4.4 Multi-turn recall pilot: no degradation at moderate context fill

The pilot found no meaningful degradation at this scale. Each Figure-4 cell pools 15 calls (5 needle positions $\times$ 3 trials). 35 of the 36 cells (3 models $\times$ 4 languages $\times$ 3 fill levels) scored a perfect 15/15; the one exception, GPT-3.5 on Malayalam at 100% fill, scored 14/15 (0.93). Every model stayed at 0.93 or higher recall across all languages and fill levels.

This matches what the window-dependent risk score (§3.3) predicts. At context sizes inside a model’s window (up to about 12k tokens here, against 16k–128k real windows), the tokenizer cost compresses the budget (Figures 2–3) but does not yet cause recall loss. Prior long-context work locates degradation near or beyond the window; our pilot is consistent with that, but it does not itself show where degradation begins. Reaching that regime requires pushing context to the window edge (32k–128k), which the runner supports via `--base-window`, and is the next experiment. One caveat cuts the same way: the Latin marker may stand out in a non-Latin haystack, which would make retrieval *easier* for high-cost languages, not harder, so a script-native marker is a planned variant.

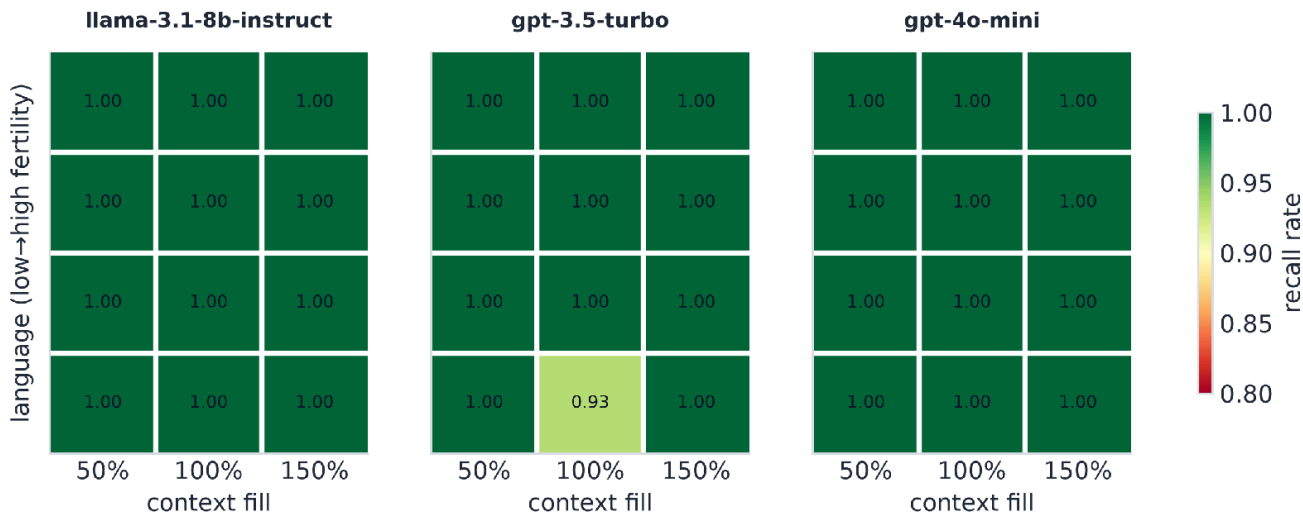


Figure 4. Measured needle recall; each cell pools 15 calls (5 positions $\times$ 3 trials), 540 calls total across 3 models $\times$ 4 languages $\times$ 3 context-fill levels.

5. Discussion and Limitations

The tokenizer cost is a concrete, measurable case of a single Global-North-centred default producing unequal outcomes: the same model and prompt, several times the cost and a fraction of the usable context for already-under-served languages, compounding with use. Our results reframe it from a fixed property of “hard languages” into a choice a buyer can change: for the worst-hit scripts, a better-matched tokenizer cuts the cost by 3–4 $\times$, and a team can check this before deployment and re-run it when the model, corpus, or prices change.

This connects to a broader point about safety evaluation. An average score can hide a per-language failure: a model with a high average harmlessness score can fail almost entirely in one language, as the cross-lingual jailbreak result above shows. Safety and cost benchmarks alike should be reported per language and per effective context, and FertiScope is a fast first-cut before a full SEA-HELM-class evaluation. Because the analyzer runs in the browser with no API call or GPU (the §3.4 benchmark is a separate offline script), a team in a compute-constrained setting can run the check on its own corpus for free.

A note on Vietnamese. Hosting this work in Da Nang invites the question of how the Vietnamese language performs. We found Vietnamese is among the least-taxed languages in our set because its Latin-with-diacritics orthography is well covered by modern vocabularies: 2.25× English on the GPT-4/3.5 tokenizer and 1.36× on Llama-3.1/SEA-LION. This is in line with a broader finding in the research that cost tracks how well the tokenizer covers the script. So FertiScope helps a Vietnamese deployer less for Vietnamese itself and more for the region’s costly scripts, Khmer (8.3×), Lao (8.6×), and Burmese (11.2×), where a tokenizer choice moves the bill several-fold.

Limitations

- **No causal multi-turn claim.** The pilot found no recall loss at up to 12k tokens (§4.4), and in our research plan we conclude that the cross-lingual fertility-to-degradation effect is suggestive, but not identified. The reasons for this are that several: fertility is hard to separate from training-data scarcity for Tamil and Burmese, a frozen-weight tokenizer swap cannot cleanly separate the two, and degradation appears near or beyond the window, not at the fills we tested. Therefore in this work we make no accuracy-loss prediction.
- **Domain dependence.** The leaderboard reflects FLORES news/wiki text. Absolute numbers shift by domain and register. This is addressed by the Analyzer running on the user’s own text.
- **Tokenizer scope.** We test the tokenizers of GPT-4o, GPT-4/3.5, and Llama-3.1 (which SEA-LION v3 uses unchanged). We do not test SeaLLMs 3, which is built on a different, Llama-2-based tokenizer with an extended vocabulary that would have different cost implications.
- **Metric scope.** Fertility alone is an incomplete efficiency metric (see Nayeem et al, 2025). In the literature review we found that STRR and Rényi efficiency are not yet reported.
- **Sample size.** This work is only intended to be a pilot study. Therefore the leaderboard of Fertiscope uses 50 parallel FLORES sentences out of the whole dataset. While tokenization is deterministic per string, 50 sentences is small for language-wide claims. Using the full FLORES split with per-sentence variance would firm up the absolute ratios. The Fertiscope Analyzer sidesteps this by running directly on the user’s own text.
- **Segmentation.** For spaceless scripts, per-word fertility depends on the segmenter; we fix and disclose Intl.Segmenter. The headline per-sentence cost ratio does not depend on it.

Future Work

The literature places the context window degradation effect closer to window capacity. Therefore future work should focus on this, and run the planned 10-turn cumulative-QA benchmark with its controls.

Future work should also add STRR and Rényi efficiency, more tokenizers (Gemma, Qwen, and purpose-built Indic tokenizers such as IndicSuperTokenizer), corpus upload, and a per-language safety-floor view.

6. Conclusion

We find that the multilingual LLM cost is real, large, and fixable based on choice of tokenizer. Expanding on this result, we built the web tool called FertiScope which reports the cost in dollars and usable context for 15 lower-resource Asian languages across three production tokenizers.

This tool shows that the worst impacted scripts (Burmese, Sinhala, Kannada, Malayalam, Telugu, Tamil) can be relieved by 3–4× with one tokenizer decision, while today’s regional models change the weights but not the tokenizer for those scripts. In this work we also draw a line between the measurements and the estimates, so the tool can inform a deployment choice without overstating the harm.

Code and Data

- **Code repository:** <https://github.com/dangleo-ctrl/fertiscope>
- **Live demo:** <https://fertiscope.vercel.app>
- **Data:** FLORES-200 (No Language Left Behind, Meta AI) — CC-BY-SA 4.0; precomputed 16-language (English plus 15 Asian languages) × 3-tokenizer leaderboard in the repo (data/leaderboard.json, regenerable via scripts/precompute.mjs).
- **Stack:** Next.js 16 · React 19 · Tailwind v4 · Recharts; pure-JS tokenizers (js-tiktoken, llama3-tokenizer-js).
- **Figures and benchmark:** research/ holds make_figures.py (all four figures), benchmark_runner.py (the §3.4 pilot), the measured data/benchmark_results.csv (540 calls), and a per-call log; see research/README.md.
- **Reproducibility.** The leaderboard, cost, capacity, and context-budget numbers are deterministic functions of public tokenizers over public FLORES-200 text and reproduce exactly from the repo; the §4.4 recall numbers are measured API outputs, logged per call. The engine generalises to any language and to any tokenizer with a JS encoder.

Author Contributions

Leo Dang led the literature review and exploratory research direction. Antoine Pedretti carried out engineering and platform work for the experiments and the deployed tool. Miles Whiticker documented the experimental design, results, and write-up. Vinh Van contributed to research and review. All authors reviewed the final report.

References

- Ahia, O., Kumar, S., Gonen, H., Kasai, J., Mortensen, D. R., Smith, N. A., & Tsvetkov, Y. (2023). *Do all languages cost the same? Tokenization in the era of commercial language models*. arXiv. <https://doi.org/10.48550/arXiv.2305.13707>
- Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, V., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Gong, L., Edunov, S., & the NLLB Team. (2022). *No language left behind: Scaling human-centered machine translation*. arXiv. <https://doi.org/10.48550/arXiv.2207.04672>
- Laban, P., Hayashi, H., Zhou, Y., & Neville, J. (2025). *LLMs get lost in multi-turn conversation*. arXiv. <https://doi.org/10.48550/arXiv.2505.06120>
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). *Lost in the middle: How language models use long contexts*. *Transactions of the Association for Computational Linguistics*, 12, 157–173. https://doi.org/10.1162/tacl_a_00638
- Lundin, J. M., Zhang, A., Karim, N., Louzan, H., Wei, V., Adelani, D., & Carroll, C. (2025). *The token tax: Systematic bias in multilingual tokenization*. arXiv. <https://doi.org/10.48550/arXiv.2509.05486>
- Nayeem, M. T., Alqahtani, S., Laskar, M. T. R., Mohiuddin, T., & Bari, M. S. (2025). *Beyond fertility: Analyzing STRR as a metric for multilingual tokenization evaluation*. arXiv. <https://doi.org/10.48550/arXiv.2510.09947>
- Ng, R., Nguyen, T. N., Huang, Y., Tai, N. C., Leong, W. Y., Leong, W. Q., Yong, X., Ngui, J. G., Susanto, Y., Cheng, N., Rengarajan, H., Limkonchotiwat, P., Hulagadri, A. V., Teng, K. W., Tong, Y. Y., Siow, B., Teo, W. Y., Lau, W., Tan, C. M., Ong, B., Ong, Z. H., Montalan, J. R., Chan, A., Antonyre, S., Lee, R., Choa, E., Ong Tat-Wee, D., Liu, B. J. D., Tjhi, W. C., Cambria, E., & Teo, L. (2025). *SEA-LION: Southeast Asian languages in one network*. arXiv. <https://doi.org/10.48550/arXiv.2504.05747>
- Petrov, A., La Malfa, E., Torr, P. H. S., & Bibi, A. (2023). Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems* (Vol. 36). <https://doi.org/10.48550/arXiv.2305.15425>
- Sayeedi, M. F. A., Alam, M. M., Rahman, S. S., Islam, M. A., Deepti, J. F., Mohiuddin, T., Islam, M. M., & Shatabda, S. (2025). *Ready to translate, not to represent? Bias and performance gaps in multilingual LLMs across language families and domains*. arXiv. <https://doi.org/10.48550/arXiv.2510.07877>
- Susanto, Y., Hulagadri, A. V., Montalan, J. R., Ngui, J. G., Yong, X. B., Leong, W., Rengarajan, H., Limkonchotiwat, P., Mai, Y., & Tjhi, W. C. (2025). *SEA-HELM*:

Southeast Asian holistic evaluation of language models. Findings of the Association for Computational Linguistics: ACL 2025. <https://doi.org/10.48550/arXiv.2502.14301>

Vázquez, R., Mickus, T., Zosa, E., Vahtola, T., Tiedemann, J., Sinha, A., Segonne, V., Sánchez-Vega, F., Raganato, A., Libovický, J., Karlgren, J., Ji, S., Helcl, J., Guillou, L., de Gibert, O., Bengoetxea, J., Attieh, J., & Apidianaki, M. (2025). *The multilingual shared task on hallucinations and related observable overgeneration mistakes (Mu-SHROOM)*. arXiv. <https://doi.org/10.48550/arXiv.2504.11975>

Yong, Z.-X., Menghini, C., & Bach, S. H. (2023). *Low-resource languages jailbreak GPT-4*. arXiv. <https://doi.org/10.48550/arXiv.2310.02446>

Zhang, W., Chan, H. P., Zhao, Y., Aljunied, M., Wang, J., Liu, C., Deng, Y., Hu, Z., Xu, W., Chia, Y. K., Li, X., & Bing, L. (2024). *SeaLLMs 3: Open foundation and chat multilingual large language models for Southeast Asian languages*. arXiv. <https://doi.org/10.48550/arXiv.2407.19672>

Appendix

A. Multi-turn risk score. Let t_{75} be the turn at which cumulative tokens cross 75% of the window. $\text{budgetScore} = 2$ if $t_{75} < 6$, 1 if $t_{75} < 12$, else 0. $\text{fertilityRaw} = 2$ if the fertility ratio ≥ 8 , 1 if ≥ 4 , else 0. $\text{fertilityScore} = \text{fertilityRaw}$ only when $\text{budgetScore} > 0$, else 0. $\text{Total} = \text{budgetScore} + \text{fertilityScore}$ in $[0, 4]$, mapped to Low (≤ 1), Medium (2), High (≥ 3). The amplifier rule is why a 128K window stays Low for short chats regardless of fertility.

B. Cost-model parameters (editable in-app, representative 2026 prices, \$/1M in/out): GPT-4o \$2.5/\$10 (o200k); GPT-4 Turbo \$10/\$30 (cl100k); GPT-3.5 Turbo \$0.5/\$1.5 (cl100k); Llama-3.1-8B hosted \$0.2/\$0.2 (llama3).

C. Fertility tiers (tokens/word): Excellent < 1.6 · Good < 3 · Moderate < 6 · High < 10 · Severe ≥ 10 .

D. Figures. All figures regenerate from `research/make_figures.py` over the bundled FLORES-200 leaderboard and the benchmark results. Figures 1–3 are deterministic; Figure 4 is rendered from the measured `research/data/benchmark_results.csv` (the 540-call pilot). PNG (300 dpi) and vector PDF are emitted alongside the scripts.

LLM Usage Statement

We used LLM assistance throughout. The literature review and the underlying deep-research report were produced with multi-agent LLM workflows (Claude Code; GLM-5.2 via OpenRouter), and LLMs assisted in drafting and editing this report. The quantitative cost results are deterministic outputs of the FertiScope tool (real tokenizers over FLORES-200) and reproduce from the public repository; the \$4.4 recall figures are measured model outputs, logged per call. Citations were verified against arXiv and the ACL Anthology (June 2026); two author attributions were corrected and one unverifiable per-language statistic was removed. We report no executed result we did not run, and we do not make the causal multi-turn claim the evidence does not support.