When the Window Cannot See the Question: Token-Denominated Observation Windows Are Not Language-Neutral

Anonymous authors
Paper under double-blind review

Abstract

KV-cache evictors of the SnapKV family decide what to keep by scoring the cache from the last w tokens of the prompt alone. When those tokens do not hold the question, the ranking is blind: it keeps the question's own keys and drops most of the gold passage's. A padded instruction, a JSON schema and a tokenizer each put the question outside the window, and the tokenizer is the lever an English-validated benchmark cannot see. On Qwen3-4B one line of instructions and the chat suffix leave 25 tokens after the question in English, 107 in Bengali and 167 in Telugu, so a research library's default w=64 leaves English untouched and costs Bengali and Telugu double digits. The fix is a check, not a tuned constant: $\hat{w} = c + Q_{90}$, the trailing block plus a question-sized slack, both measured before any forward pass, says whether a shipped window is safe for a prompt family, returns the smallest that is, and says when none is. It removes the blind mode and restores the passage, $+14$ over the default on the full Telugu pool at a tenth more prefill time; for every public static tail we found, a fixed 256-token window does as well. Neither removes the eviction cost a fully visible question still pays. Because no window costs English anything at the shipped ratio, an English-validated benchmark cannot tell the two shipped constants apart; the window is one token-denominated constant among several, and the check generalises: measure the constant on the tokenizer before trusting an English validation.

1 Introduction

Observation-window evictors such as SnapKV (Li et al., 2024) decide what to keep in the key–value cache by scoring every cached key against the last w tokens of the prompt and discarding the rest. In a retrieval prompt laid out as `passages + question + instruction`, whether the window reaches the question is decided by one integer, c, the tokens between the question and the end of the prompt: if $c > w$ the ranking is blind (Figure 1).

A blind ranking does not evict the question. It keeps the question's keys and drops most of the gold passage's, and that is the loss that matters. The window is a threshold, and where it sits is geometry: how many tokens the instruction and the chat suffix occupy after the question. Tokenizers move that geometry per language: the same one-line instruction is $1.15\times$ English in characters on Telugu and $8.1\times$ in tokens, because the vocabulary reaches the script only at the byte level (Section 3). On Qwen3-4B that line plus the suffix leaves 25 tokens after the question in English, 107 in Bengali and 167 in Telugu, so the research default w=64 (kvpress, Jégou et al., 2024) — opt-in at most in production (Appendix U) — leaves English prompts intact and blinds Bengali and Telugu, which lose double digits on their full validation pools (Section 6). In the library's own pipeline, which hides the question from the window in every language, every language pays (Appendix O); single-string prompting is where the window works, and where the tokenizer decides for whom. What is compared across languages is where the threshold sits; the depth of the loss past it is confounded with passage length and is not (Section 8).

Four levers move c past the window, and language is only the tokenizer's. Padding the English instruction until c passes w moves the threshold with the pad without leaving English. An English JSON response schema puts its trailing block one token from Telugu's and is blinded the same way.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A tokenizer that encodes the Bengali instruction in 26 tokens rather than 102 moves the cliff along with it (Section 6). And every turn of an agent loop appends to the tail, so public scaffolds pass any constant by their second turn; that geometry we measure, the loss we do not (Appendix T).

That a window can miss a tail is true by construction. The size of the loss is not: double digits against zero, from one line the tokenizer makes five to eight times longer. Nor are three things past the threshold. Clearing it by a few tokens recovers nothing: a question-sized slack is the smallest that clears the residual we can detect, and query visibility, $V = |W \cap Q|/|Q|$, computable from the tokenizer and the template with no forward pass, is the dose (Section 5). A fully visible question still pays an eviction cost that no window removes (Section 6). And the threshold is visible at the key level: the blind ranking keeps the question and loses part of the passage, a wider window buys some back, and what turns retention into accuracy is left open (Section 7).

The measurement implies a rule measured rather than tuned: $\hat{w} = c + Q_{90}$, the trailing block plus a question-sized slack from held-out questions. It removes the blind mode and restores the passage — $+14$ and $+13$ over the default on the full Telugu and Bengali pools — and returns the shipped constant where that suffices. It does not make eviction free: Telugu keeps a -5.7-point residual that a per-item oracle window does not remove, and five further models miss the gate. Whether the fix is an English tail or a sized window, the same measurement comes first, and it is not the window's alone: any constant denominated in tokens — a window, a decode cap (Appendix P), a compression boundary (Appendix O) — is set per language by the tokenizer, and nothing in an English-validated benchmark says which one overflowed.

Contributions. (i) A threshold in c: a failure of KV eviction localised to one integer, moved by four levers — pad, schema, tokenizer, turn — with language held fixed or varied, and observed at the key level. (ii) A dose: query visibility V from the input alone, the question-sized slack it implies (a fixed slack of 32 or more lands about two points short of it), and the eviction cost that remains past $V{=}1$. (iii) A safety check, not a remedy: two measurements say whether a shipped window is safe for a prompt family, return the smallest that is, and say when none is; a fixed 256 is safe for every static tail we found and no window clears an agent loop. Every miss is reported with its interval, and the diagnosis is that an English-validated benchmark cannot tell the two shipped constants apart, which is how the number can go unaudited.

Scope. The named default is a research library's; the production stacks we checked expose no such evictor by default (Appendix U). This is a mechanism study of that default and a check for any caller who applies a last-w scorer to a single-string prompt with a long tail after the question — translated instructions, schemas, scaffolds — not an audit of production stacks. No public multilingual long-context suite carries a translated tail after the question, and the one that carries a localised tail retains its needle (Appendix T).

2 Related work

Observation windows and query visibility. Prefill eviction scores the cache from a last-w slice of the prompt (Li et al., 2024); the attention SnapKV accumulates inside its window and H2O (Zhang et al., 2023) over the whole prefill, and the estimated attention of Devoto et al. (2025) are computed from the model, V from the tokenizer and the template alone. Table 1 places this paper against the two closest results.

Table 1: What the two closest results show, and what this paper adds.

	shows	leaves open
Luo et al. (2026)	question visibility reorders compression methods on English benchmarks	a continuous, forward-pass-free measure; who is exposed, by how much
Chen et al. (2026)	eviction discards instructions in English prompts; a whitelist repairs it	the window as a threshold; a dose; a remedy without prompt structure
this paper	V from the input; the threshold at c, moved with language fixed; the dose; a safety check	the eviction cost past $V{=}1$; iterated agent prompts

How w gets chosen. Recent eviction work treats w as a hyperparameter to set rather than a threshold to test: IntentKV (Zhao et al., 2026) and the original SnapKV retrieval configuration use w=64, and Nawrot et al. (2026) sweep windows to a task-average optimum. The window descends from the sink-plus-window cache of StreamingLLM (Xiao et al., 2024); Quest (Tang et al., 2024) scores pages against the decode query, so V=1 there by construction. Ada-KV, DuoAttention, KVzip and ChunkKV (Feng et al., 2024; Xiao et al., 2025; Kim et al., 2025a; Liu et al., 2025) ship a budget or a window as a constant, none tested against a trailing instruction of varying length.

Instruction order, protection, and delimiters. Chen et al. (2026) document that eviction can discard an instruction in multi-instruction English prompts and propose a whitelist; their reordering moves two instructions inside a system prompt; we move the instruction relative to the question and grow it until it overflows the window, with a fix that needs no knowledge of prompt structure. Liu et al. (2023) show that instruction position changes generation quality; here it changes what the scorer sees. Garcia (2026) shows that protecting a prompt's prefix and suffix outperforms scoring under a global cache cap, assuming the critical tokens occupy a fixed 15–30-token budget; the Bengali and Telugu instructions occupy 102 and 162, and we test that assumption. Protection with a measured budget is this paper's rule. FINCH (Corallo & Papotti, 2024) sizes its window from a user-supplied delimiter, and question-conditioned prompt compression (Jiang et al., 2024) scores tokens against the question itself, so V=1 holds in both by construction — evidence that w must adapt, by a route that presupposes a question the caller can pass. Sclar et al. (2024) and Mizrahi et al. (2024) show that formatting alone moves accuracy by several points; the blind mode is a compression-side case with the mechanism located.

Tokenizer fertility. Petrov et al. (2023) and Ahia et al. (2023) established that a fixed token budget is not language-neutral; subsequent work studied the cost, latency and usable context that follow, including under weight quantization (Marchisio et al., 2024; Marie & Fujita, 2025), budgets the user can see. Multilingual prompting found English instructions competitive for non-English inputs (Shi et al., 2023); Appendix N is that choice's effect on the window. The constant studied here sits inside the scorer; to our knowledge no eviction study attributes per-language damage to it.

3 SETUP

Task and data. Every item is a question, its gold passage, and same-language distractor passages packed to a fixed 8k-token prefill, gold position rotated by item index. Questions and passages come from XQuAD (Artetxe et al., 2020) for English, Chinese, Spanish, Vietnamese and Thai, and from TyDiQA-GoldP (Clark et al., 2020) for Swahili, Bengali and Telugu. The two collections are not comparable, so we never compare accuracy across languages. Every number in this paper is a within-language, item-paired difference between two configurations evaluated on the same items: 100 per cell, except where Section 6 pairs the entire validation pools (669 Telugu, 113 Bengali) and marked supporting runs pair 200. Each comparison stays inside one model and one software stack; follow-up runs repeated 10,301 earlier generations byte for byte (Appendix R).

Models and compression. Qwen3-4B (Qwen Team, 2025) is the primary model; Qwen3-8B is a scale slice and Gemma-3-4b-it (Gemma Team, Google DeepMind, 2025) a tokenizer contrast; Llama-3.1-8B, Qwen3-32B and Gemma-3-12b-it enter in Section 6. We compress with SnapKV (Li et al., 2024) through the kvpress implementation at a fixed eviction ratio r=0.75 (a 2,048-token budget at 8k, the registered midpoint) and expose the observation window as the treatment variable. The press runs over the whole prompt at prefill, question included: a serving engine receives one token sequence and SnapKV defines its window on the end of it. kvpress's context-then-question split, which scores from no question in any language, is not a safe alternative but the language-flat worst case: on the same items it costs every language 23–42 pp, English 29 (Appendix O). SnapKV keeps the window inside the retained budget, so every comparison below is at matched retained KV. Decoding is greedy with a 384-token cap (the common 128-token cap is itself a token-denominated constant; Appendix P).

Scoring. An answer counts as correct if a gold span appears, after Unicode normalization and language-aware tokenization, either inside the marked answer span (the instruction asks for #### followed by the span) or inside the first sentence of the prose before the marker, a sentence ending at

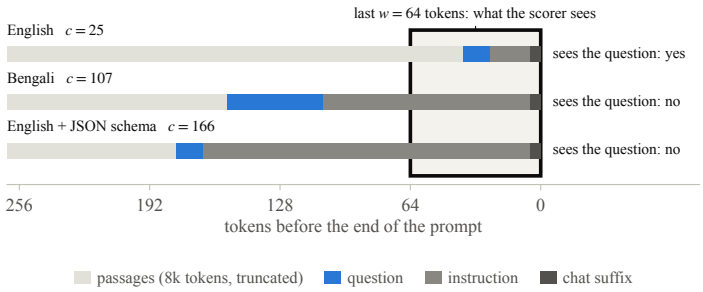


Figure 1: The instruction-last layout, drawn to token scale. Shaded: the last w=64 tokens, the only part of the prompt a SnapKV-style scorer reads. English leaves 25 tokens after the question, so the window covers it; Bengali leaves 107 and an English prompt carrying a JSON schema 166, and in both the scorer ranks the passages having seen none of the question. Widths are item medians; passages continue leftwards.

the first sentence-final punctuation (full stop, danda or CJK stop) or, where the script carries none, at the first line, and tokens being whitespace words, or characters for scripts written without word separators. The rule was fixed before the runs reported here and is applied offline to raw generations, with no LLM judge anywhere in the paper; a stricter, marker-only variant leaves every conclusion it re-scores unchanged (Appendix D). Paired comparisons are two-sided McNemar tests, reported with their discordant counts (fixed/broken); stars mark uncorrected $p < .05$. The one grid that is a family of tests (Table 2, 25 tests) is Holm-corrected in Appendix D, where five blind cells survive; no claim rests on a cell the correction removes.

Reading guide. *blind*: the window holds no token of the question (V=0). *hole*: a paired loss against the uncompressed baseline at a shipped window. *interval-cleared*: the exact 95% interval lies beyond the stated bound. *confirmed*: the interval excludes zero. *gate met*: the preregistered ±3 pp point rule passed; at n=100 the interval stays wide.

The quantities. Prompts are instruction-last unless stated: `passages + question + instruction + chat suffix`. Write I for the token length of the trailing instruction, s for the chat-template suffix after it, c=I+s for the tokens after the question, Q_i for the token positions of item i's question, and W for the last w positions. The scorer ranks key j by attention $A_{t,j}$ from the window alone, query visibility is the share of the question inside it, and the rule is the window that holds a question-sized slack past the tail:

$$s_j = \sum_{t \in W} A_{t,j}, \quad V_i = \frac{|W \cap Q_i|}{|Q_i|} = \mathrm{clamp}\big(\tfrac{w-c}{|Q_i|}, 0, 1\big), \quad \hat{w} = c + Q_{90}, \text{ safe when } \hat{w} \le w. \quad (1)$$

If $W \cap Q_i = \emptyset$, no s_j depends on the question: the ranking is a function of the passages and the tail alone, whatever the model. On Qwen3-4B the eight instructions, written in English and translated by the authors at 93–126 characters each (41 in Chinese), tokenize to I between 20 (English) and 162 (Telugu): the vocabulary reaches Bengali and Telugu script only at the byte level, 59% and 81% of their tokens being sub-character pieces, none for the other six or on Gemma's vocabulary (Table 12); with s=5 this gives c from 25 to 167. Both are produced by script from the run's own tokenizer and template (Appendices C and Q).

The held-out percentile. Q_{90} is measured on a held-out split that excludes the evaluation items (Appendix C), giving 18/76/80 for English, Bengali and Telugu on Qwen3-4B; it is never tuned against accuracy. Besides the template, the check needs a sample of the family's questions (here a held-out split; in a deployment, a sample of its traffic with the question located offline, once).

4 The window is a threshold

A window sweep across five languages. Table 2 and Figure 2a report paired damage against the uncompressed baseline for five languages at five windows, ratio 0.75; registered: BLIND when $w < c$, SAFE when $w > c$.

Table 2: Sweeping the observation window. Paired Δpp against the uncompressed baseline, Qwen3-4B, n=100 items per cell, eviction ratio 0.75. Shaded cells were registered BLIND ($w < c$) before the run; stars mark McNemar $p < .05$: the design predicts the shading, the outcome supplies the stars. The w=64 column comes from the AutoWindow run of Section 6 (same stack, baselines byte-identical to this sweep's) and pairs within that run.

	c	baseline	w		default	w		
			32	56	64	88	120	176
en	25	93	+2	+1	+0	+1	−1	+1
th	45	85	−6	−4	+0	+0	−2	−3
sw	47	56	−7*	−4	−3	−2	−4	−3
bn	107	73	−13*	−15*	−16*	−21*	−8	−3
te	167	56	−19*	−21*	−19*	−18*	−14*	−13*

English, at either shipped constant under a windowed scorer, shows no detectable change (exact intervals within five points) — not because every tested window holds its whole question (w=32 leaves seven tokens of slack against questions of 6–24) but empirically: partial visibility without damage recurs on Gemma (Section 6). Bengali is damaged in every blind cell (−13 to −21 pp, all $p < .01$) and comes back to −3 pp only at w=176 (c+69). Telugu is down double digits in all five cells, interval-cleared at three under Holm and directionally consistent at five. Thai and Swahili dip only at the one tested window below their c. The three languages not swept (Chinese, Spanish, Vietnamese; c=29/35/39) sit at +0, −2 and −5 at w=64 (Appendix F): Chinese, whose tokenizer is efficient, is the non-English control the account predicts safe. The rule does not make the hole: marker-only and token F1 give the Telugu cells −14 to −22 and −10 to −18; Bengali answers before its marker, so its strict rule is F1, −4 to −10 (Appendix D).

The same threshold, constructed in English. If the mechanism is trailing length, padding an English instruction should move the threshold with the pad, and widening w past the new c should clear it. We pad the English QA instruction with content-free filler to four lengths ($c = 57/73/105/137$) and sweep five windows on the same English items (Figure 2b; Table 14).

The threshold moves with the pad: at every pad the first window above c sits at its own baseline (0, 0, −1, 0 pp; Table 14), and w=32 below c is an interval-cleared loss at every pad (−11, −8, −12, −13). Against the slack $w - c$ the four conditions fall on one curve (Figure 2b); a post hoc logit on the slack finds no pad effect (p=.13; Appendix F). What the baselines add is the shape inside the blind region: blind windows of 56–104 tokens sit within noise of baseline (−2 to −6, none interval-cleared), where Bengali's blind cells stay at −13 to −21 through w=88 — blindness sets where the loss ends, not its size. Across these curves only the tokens between the question and the end of the prompt changed, and the data source does not predict the hole (Swahili, TyDiQA, is safe at 64 and damaged at 32).

A constant that is already shipped. kvpress ships w=64; the opt-in RocketKV path in TensorRT-LLM ships w=32. Moving from the first constant to the second, on matched items, costs Thai 4.5 pp (2/11, p=.022) and Swahili 6.5 pp (1/14, p=.001; n=200 — the same 100 questions at two context budgets, so these p-values are optimistic). Both languages sit above the threshold at w=64 and below it at w=32 (only Swahili's p=.001 would clear a Holm correction; Appendix D): the failure is not confined to the two scripts with the longest instructions but reaches any language whose trailing block outgrows whichever constant is in force. Thai and Swahili would take $\hat{w} = 91$ and 67, both above their own c, so the rule cannot produce this flip, and a preregistered run confirms both meet the ±3 pp gate there (Thai −1 pp, interval within −3; Swahili −3 pp, at the point rule's boundary, interval wide). Moving the instruction to the front recovers Bengali 13 pp and Telugu 14 pp on the same items (Table 3): a lever on the mechanism, not a recommendation.

5 How much slack, and in what unit

How far past the trailing block must the window reach? Telugu at c+9 was still 13 points down, so "clear c" is not the answer.

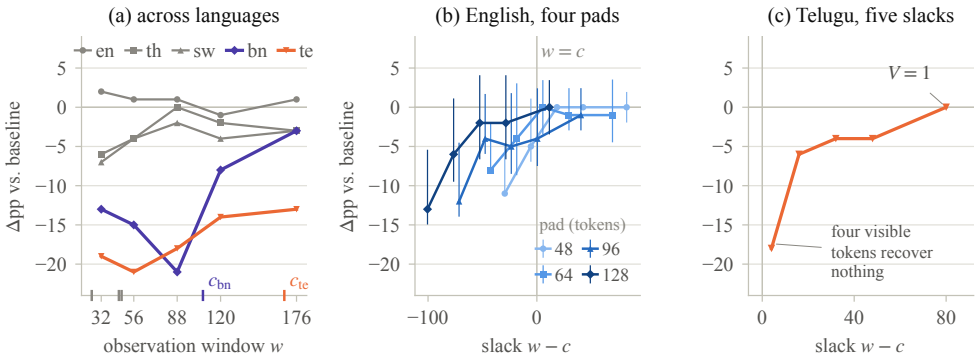


Figure 2: Three sweeps of the same mechanism, each paired against its own uncompressed baseline. **(a)** Five languages against the window; ticks mark each language's c. **(b)** English, four instruction pads, against the slack $w - c$, each against its own baseline (91–94%; bars are exact 95% intervals, offset by three tokens per pad for legibility): one curve, at baseline from the first window past $w{=}c$. **(c)** The Telugu dose ladder on its own slack range: English is at its ceiling within about ten tokens of slack, Telugu needs eighty.

A dose ladder. We sweep five windows at fixed offsets above the measured c on Telugu: $c{+}4$, $c{+}16$, $c{+}32$, $c{+}48$ and $c{+}80$, on 100 items with their own uncompressed baseline (Figure 2c, Table 15, Appendix G; offsets fixed before the run).

Four visible tokens of question buy nothing: $c{+}4$ loses 18 pp, near the $w{=}64$ loss on the same items (-19); the curve then rises with the slack and reaches the baseline only at $c{+}80$. The same ladder on Bengali, with its own c and baseline, reproduces it rung for rung (-17, -7, -3, -5, -2 pp).

Visibility as the dose. Question length varies across items (34–122 tokens), so one window spreads visibility across items. Pooling the five treated cells and binning by per-item V: items with $V < 0.25$ lose 16.1 pp ($n{=}118$, $p{<}10^{-3}$), while items with $V{=}1$ show no detectable effect ($+0.7$ pp, $n{=}135$; 9/8 discordant). Across the ladder, a logistic regression of correctness on V with item-clustered standard errors gives $+0.68$ ($z{=}3.35$, $p{<}.001$); within items, 21 switchers are correct at high visibility against 2 in the opposite direction ($p{<}10^{-4}$).

The bins are not monotone: the $0.75 \le V < 1$ band loses 11.9 pp ($n{=}59$, $p{=}.065$) and the Bengali ladder breaks in the same band, both on long-question items that recover at $V{=}1$; we claim the window-level curve and the within-item trend, not a monotone bin profile.

6 AutoWindow: one integer, and where it stops

The rule. Both quantities are available before any forward pass, and the check returns one of three answers: the shipped window is safe ($\hat{w} \le w$), the smallest safe window is $\hat{w}$, or none is safe because the tail is not a constant of the template:

> **AutoWindow** — one integer per (model, language, template), no forward pass, no delimiter
> 1. $c \leftarrow$ tokens after the question in the template: instruction + chat suffix (`measure_c.py`).
> 2. $Q_{90} \leftarrow$ 90th percentile of question length on a held-out split (`measure_q.py`).
> 3. $\hat{w} \leftarrow c + Q_{90}$, the observation window at the chosen eviction ratio.
> 4. If the template has no fixed tail — an iterated prompt appends to it every turn — no window clears it: protect or reorder.

It buys no cache and costs time: peak memory is identical at 64, 247 and 256 (10.94 GB) and prefill rises from 1.07 to 1.18 s at Telugu's 247 on one A6000 with kvpress's unoptimised scorer, whose share goes from 6.5% to 15% (Table 8). A per-item window $w_i{=}c + |Q_i|$ would give $V_i{=}1$, but a preregistered full-pool run of that oracle buys almost nothing (below) and needs a question-locating parser on the serving path (in a single-turn retrieval prompt the chat template's role boundary does not supply one; Appendix E).

Table 3: AutoWindow at eviction ratio 0.75. Accuracy, with paired Δpp against each block's own uncompressed baseline in parentheses; n=100; star: McNemar $p < .05$. (A) Qwen3-4B on three of the eight languages; (B) English prompts carrying a JSON response schema padded to 60, 120 and 200 tokens, c measured with the schema in place. The last column is the layout remedy — the question moved to the end of the prompt, V=1 by construction — against that layout's own baseline. The full-pool re-run is reported below and in Appendix E.

	window		accuracy (Δpp vs. baseline)			layout
	c	$\hat{w}$	uncompressed	w=64	at $\hat{w}$	Δpp
(A) languages						
en	25	43	93	93 (+0)	**95** (+2)	95 (−1)
bn	107	183	73	57 (−16*)	**71** (−2)	70 (−4)
te	167	247	56	37 (−19*)	**56** (+0)	51 (−6)
(B) English with a JSON schema tail						
JSON-60	72	90	94	88 (−6)	**96** (+2)	—
JSON-120	166	184	95	89 (−6*)	**96** (+1)	—
JSON-200	213	231	94	89 (−5)	**96** (+2)	—

The registered gate. Panel (A) of Table 3 is the preregistered test ($|\Delta| \leq 3$ pp against the uncompressed baseline); the formula meets it at n=100 on both languages. On the entire validation pools (preregistered; Table 11) Bengali holds the gate and Telugu does not: -5.7 pp (n=669, $[-7.8, -3.1]$) against a -20.2 pp default hole; the new items sit at -6.7, and the full-pool number is the headline.

The percentile, separated from a constant slack. A preregistered ablation re-ran the entire Telugu pool at c+16, c+32 and $\hat{w}$ in one run: each fixed slack leaves a strictly deeper interval-cleared residual (-10.3 and -7.8 against -5.7); the registered head-to-head separates the rule from the English-sized slack by $+4.6$ pp (55/24, p=.0006) and from c+32 by $+2.1$ with zero inside the interval. A second run put the median question beside the ninetieth percentile: $c+Q_{50}$ leaves -7.6, a difference of $+1.9$ ($[+0.1, +3.5]$; Appendix E). The unit, a question, carries most of the effect; the percentile inside it is worth about two points, and Q_{90} is the registered choice that never trailed.

Schema tails. The same logic applies when the tail is a response format (a mis-sized run: Appendix H). Measuring c with the schema in place gives 72, 166 and 213 tokens, hence $\hat{w} = 90$, 184 and 231. Panel (B) shows the formula closing all three tails (8/0, 7/0, 7/0 fixed/broken vs w=64). The 120-token schema's trailing block, 166 tokens, sits one token from Telugu's 167: the same blindness from a response format instead of a script, removed by the same integer.

A second tokenizer. Gemma-3-4b-it tokenizes the Bengali instruction to 26 tokens against Qwen's 102 and its questions to a median of 11: c=27/32/44 and Q_{90}=18/18/20 for en/bn/te, giving $\hat{w} = 45/50/64$. At 4B the rule returns the shipped constant for Telugu (gate met, -3) and leaves Bengali on a plateau ($\hat{w}$ -6, default -5): no blind mode to remove. At 12B, where Telugu's baseline is 61, the blind w=16 loses -7 ($[-8.9, -0.3]$) and Bengali's $\hat{w}$ recovers $+8$ over it ($[+0.4, +11.5]$; Appendix I).

Scale, prefill, and a third family. The hole reproduces, in at least one language, on every model and prefill we ran; the rule's sufficiency does not (Table 4). At Qwen3-8B and on Llama-3.1-8B (a third tokenizer family, $\hat{w}$=212 for Bengali) the default costs Bengali 17–18 points and $\hat{w}$ recovers +9/+10 while remaining 8 below baseline; at Qwen3-32B the holes persist on both languages and $\hat{w}$ leaves -1.6 on the full Telugu pool with zero inside the interval; at twice the prefill the *same* $\hat{w}$=247, prefill-free by construction, meets the gate.

What the residuals are not. Eight cells miss the gate: Bengali at Gemma (-6), 8B and Llama (-8) and 32B (-5), Telugu at Gemma-12B (-5) and the 4B full pool (-5.7, interval-cleared), and both at matched passage length (te -6; bn -9, confirmed, where the rule recovers $+8$ of a -17 hole; Appendix J). None is mainly a visibility failure. With the question last every window sees all of it by construction, and a preregistered instruction-first re-run of the entire Telugu pool shows

Table 4: The hole and the remedy beyond the primary model. Δpp against each run's own baseline; n=100 unless the pool is stated; brackets are exact 95% intervals; Table 11 carries the rest.

model	language	default w=64	$\hat{w}$	$\hat{w}$ vs default
Qwen3-8B	bn	-17^{*}	-8	$+9^{*}$
Llama-3.1-8B	bn	-18^{*} $[-21.5, -9.2]$	-8	$+10^{*}$
Qwen3-32B	bn	-12^{*}	-5	$+7$
Qwen3-32B	te, 669	-10.2 $[-13.1, -6.8]$	-1.6 $[-3.5, +0.4]$	$+8.5$ $[+5.4, +11.1]$
Qwen3-4B, 16k	te	-15^{*} $[-19.7, -5.7]$	-2	$+13^{*}$

that eviction still costs -3.4 pp there ($[-5.3, -1.2]$). A preregistered per-item oracle window (V_i=1 exactly) leaves -5.1 ($[-7.3, -2.4]$), indistinguishable from $\hat{w}$ ($+0.6$, $[-1.2, +2.3]$); its residual concentrates mid-document (Liu et al., 2024), which visibility, constant across positions, cannot produce; where that residual matters, the lever left is the ratio, not the window (Table 6). Each missed cell is as rich in fully visible questions as its pool (Appendices I–L).

Nor is the integer a property of one method: dropped unchanged into PyramidKV (Cai et al., 2024), which inherits the same window, it takes Bengali from -28 to -2 and Telugu from -32 to -1 (own baselines; Appendix O). Two scorers that never read the question, Expected Attention and TOVA, lose in every language at this ratio, English included (-8 to -20; Appendix O): that is the price of ranking without the question, paid by every language alike, not a window effect; a window can be sized, query-agnosticism cannot. Table 5 sets the rule beside the other remedies: each removes the blind mode; none removes the eviction cost at this ratio, oracle included.

Table 5: Which remedy removes what. Qwen3-4B, r=0.75, Δpp against each run's own baseline; full-pool cells interval-cleared unless marked ns; n=100 cells sit within about ±8 pp.

remedy	blind mode removed?	loss after the fix, vs. uncompressed	needs
one English instruction (c=25)	yes, by construction	partial visibility: bn -3, te -4 (ns)	a tail that can be English
question last (V=1)	yes, by construction	eviction cost: te -3.4 (full pool)	control of the layout
$\hat{w} = c + Q_{90}$	yes: +14/+13 over the default at the full pools	te -5.7; bn -1.8 at 4B, -5 to -9 on other models and at matched passage length	c per prompt family, held-out Q_{90}; scorer time 6.5 → 15% of prefill
one constant, w=256	while the tail fits; -20 past it (te c=270)	ties $\hat{w}$ on bn/te at every ratio while $c < 256$; -9 on English at r=0.9375	nothing

Table 5 is the check's decision table, not a contest the rule must win, and its three answers map onto it. Where the caller authors the prompt, put the question last or repeat it at the end: V=1 by construction, no measurement, and a smaller residual than any window (-3.4 on the full Telugu pool; -7/-6 for the repeat at the default window; Appendix O). Where the tail is a constant that fits inside 256 tokens, a fixed w=256 ties $\hat{w}$ on Bengali and Telugu at every ratio we ran, so that constant is the answer for every public static tail we found; it fails where the tail does not fit (a 270-token Telugu tail: -20 against $\hat{w}$'s -6, $+14$ head-to-head; Appendix O) and costs English -9 against -1 at r=0.9375. Where the tail is not a constant, no window is safe: protect or reorder. Table 6 bounds the exposure by ratio: at r=0.5 the default's loss is inside noise; at 0.75–0.875 the rule recovers $+14$ to $+19$, interval-cleared; at 0.9375 it recovers $+8$ (ns) of Telugu's -42.

7 Analysis

What the blind ranking keeps. A preregistered prefill-only arm recorded which keys the ranking retains (Appendix J). At the shipped window it keeps 97–98% of the question's keys on Telugu and Bengali and 41–43% of the gold passage's (a distractor's: 32%); a question-sized window leaves the question at 100% and raises the gold share to 48–54% (paired $+0.07$ and $+0.11$). A blind ranking

Table 6: The window across eviction ratios (Qwen3-4B, n=100, one store, own baselines). Paired Δpp; star: McNemar $p < .05$; 0.75 is Table 3; intervals and $\hat{w}$ vs 256 in Appendix O.

	r	default w=64	$\hat{w}$	w=256	$\hat{w}$ vs default
te	0.5	-4	$+1$	-1	$+5$
	0.875	-33^*	-15^*	-13^*	$+18^*$
	0.9375	-42^*	-34^*	-35^*	$+8$
bn	0.5	-4	$+2$	$+1$	$+6$
	0.875	-30^*	-12^*	-17^*	$+18^*$
	0.9375	-44^*	-24^*	-28^*	$+20^*$

is gold-poor, not gold-blind, and across 19 blind cells on three layouts and two tokenizers the loss follows the gold share it keeps (Spearman $+0.76$): the threshold is in the input, whether the window holds the question, and the loss is continuous in what the ranking then keeps. At matched passage length the Bengali hole survives in full (-17); Telugu's cell is inconclusive (Appendix J).

Results that do not support the account. A stuffed arithmetic-reasoning control does not reproduce the effect (Bengali -3, ns; Appendix I): the failure needs a question matched against passages. Blindness is necessary, not sufficient: four low-visibility conditions show no damage, the English blind cells are graded where Bengali's are flat (Section 4), and the answerability control excludes the cheapest account of the asymmetry (Appendix I).

8 Limitations

A deployment that puts the question last, or its instruction first, has V=1 by construction and nothing here applies (instruction-first: Table 11, -3.4). Put together, the exposure needs a windowed evictor, a tail after the question longer than its window — translated instructions, schemas, scaffolds — and a ratio of about 0.75 or more: one English instruction for every language (bn -3, te -4, ns), the refine template and r=0.5 each drop one condition with no interval-cleared damage (Appendices N, T, O).

By design the percentile leaves the longest question decile below V=1, and a large w taxes prefill time ($+10\%$ at $\hat{w}$) and, at r=0.9375, the content budget (English -9; Telugu recovers $+8$, ns, of -42; Appendix O).

Our questions and passages are XQuAD's and TyDiQA-GoldP's; the 8k packing with same-language distractors is ours, because public suites vary question length, instruction length and language together, the confound we hold fixed; the one agent scaffold with a long English tail is run only as a tail on our payload, and whether eviction harms an agent loop mid-trajectory is untested here (Appendix T). The depth of the blind loss across languages is confounded with passage length; at matched length the Bengali hole survives in full and the Telugu cell is inconclusive (under a baseline that chunking cut from 56 to 34), so the Telugu regime is itself sensitive to how passages are divided (Appendix J). Finally, one primary 4B model carries the headline result; the 8B and 32B slices share its tokenizer; the third family's Telugu cell could not be scored reliably.

9 Conclusion

A token-denominated observation window is not language-neutral, because trailing instructions are not. Measuring c says whether a deployment needs to do anything at all: where the tail fits a fixed 256 the check returns it; where it does not, $w = c + Q_{90}$ removes the blind mode, leaves a residual that no oracle visibility removes, and transfers unchanged to a second evictor; where the tail grows every turn, no window clears it and the remedy is to protect or reorder — the integer belongs to the prompt, not the method. The window decides whether the ranking may see the question; what that costs is decided by what the answer needs of the gold it keeps.

ETHICS STATEMENT

This work uses publicly released question-answering corpora and publicly released model weights; it involves no human subjects and no personal data. The failure it documents falls unevenly on users who write in scripts that current tokenizers encode inefficiently, so we have tried to describe it in terms of the mechanism rather than in terms of language rankings, and to avoid comparisons of absolute accuracy between languages that our item construction does not support. The check we propose is cheap and requires no additional data collection.

REPRODUCIBILITY STATEMENT

Configurations, seeds, prompts and the scoring rule are in the supplementary code. The two constants are produced by scripts, not copied between documents (`measure_c.py`, `measure_c_schema.py`, `measure_q.py`). Every table cell is recomputed from the raw generations of the forty-nine stores listed in Appendix R by `audit_paper_numbers.py`, which fails on any drift, and comparisons never cross a stack boundary. The predictions and kill conditions of every arm are files that ship with the code, each written before its run; Appendix R records, arm by arm, which registrations carry a version-history or hosting-server timestamp that precedes the first generation and which entered version control later. The anonymised version history ships with the supplementary material; the de-anonymised history becomes public with the code.

AI USE STATEMENT

The authors reviewed all AI-assisted content and take full responsibility for every claim in this paper; this statement follows the categories of the ICLR AI policy. Large language models were used substantially and throughout this work, under three guardrails: every preregistration was reviewed, approved and committed by the authors before the corresponding run; no LLM judges any output in any reported metric; and every table cell is recomputed from the raw generation records by a committed audit script. Within those bounds, research ideation, experimental design and the preregistered predictions were developed by the authors in dialogue with LLMs. Experiments were executed by LLM agents operating the experiment harness against the committed protocols, whose kill conditions bound what an agent could decide alone. The harness, the analysis scripts and the figures are code written with substantial LLM assistance; the paper's text was drafted and revised the same way. Interpretation of results was constrained by design rather than delegated, and cross-checking between independent models during preparation surfaced and corrected errors, including hallucinated citations, caught by cross-checking and removed, and two internal miscounts, before submission. LLMs were not used to generate synthetic evaluation data; all items derive from XQuAD and TyDiQA-GoldP.

REFERENCES

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4623–4637, 2020. doi: 10.18653/v1/2020.acl-main.421.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Alex Chen, Renato Geh, Aditya Grover, Guy Van Den Broeck, and Daniel Mingyi Israel. The pitfalls of KV cache compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41530–41553, 2026. doi: 10.18653/v1/2026.acl-long.1926. ACL Anthology 2026.acl-long.1926.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. doi: 10.1162/tacl_a_00317.

Giulio Corallo and Paolo Papotti. FINCH: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532, 2024. doi: 10.1162/tacl_a_00716. Anthology 2024.tacl-1.83.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: KV cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

Gabriel Garcia. Protection is (nearly) all you need: Structural protection dominates scoring in globally capped KV eviction. *arXiv preprint arXiv:2605.18053*, 2026.

Gemma Team, Google DeepMind. Gemma 3 technical report, 2025. arXiv:2503.19786.

Simon Jégou, Maximilian Jeblick, Alessio Devoto, Jiwei Liu, and David Austin. kvpress: Llm kv cache compression made easy. `https://github.com/NVIDIA/kvpress`, 2024.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. Kvzip: Query-agnostic kv cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*, 2025a.

Yekyung Kim, Jenna Russell, Marzena Karpinska, and Mohit Iyyer. One ruler to measure them all: Benchmarking multilingual long-context language models. *arXiv preprint arXiv:2503.01996*, 2025b.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Xiang Liu, Zhenheng Tang, Peijie Chen, Hong Li, Xiuze Ma, Xiaowen Chu, Bo Xu, and Xiaowen Wang. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. *arXiv preprint arXiv:2502.00299*, 2025.

Yijin Liu, Xianfeng Zeng, Fandong Meng, and Jie Zhou. Instruction position matters in sequence generation with large language models. *arXiv preprint arXiv:2308.12097*, 2023.

Daming Luo, Christy Liang, and Junyu Xuan. How query visibility changes KV-cache compression rankings: A matched-budget audit. *arXiv preprint arXiv:2607.11942*, 2026.

Weian Mao, Yukang Chen, Wei Huang, Shuai Yang, Luozhou Wang, and Song Han. KV cache compression and its infra problems. NVIDIA Efficient AI Lab research blog, 12 June, 2026.

Kelly Marchisio, Saurabh Dash, Hongyu Chen, Dennis Aumiller, Ahmet Üstün, Sara Hooker, and Sebastian Ruder. How does quantization affect multilingual LLMs? *arXiv preprint arXiv:2407.03211*, 2024.

Benjamin Marie and Atsushi Fujita. The uneven impact of post-training quantization in machine translation. *arXiv preprint arXiv:2508.20893*, 2025.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? A call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12, 2024.

Piotr Nawrot, Jianing Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo Ponti. The sparse frontier: Sparse attention trade-offs in transformer LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 38667–38701, 2026. doi: 10.18653/v1/2026.findings-acl.1926. ACL Anthology 2026.findings-acl.1926; arXiv:2504.17768.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*, 2023.

Qwen Team. Qwen3 technical report, 2025. arXiv:2505.09388.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, 2024.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Liang Zhao, Xiaocheng Feng, Weihong Zhong, Lei Huang, Kun Zhu, Baoxin Wang, Dayong Wu, Guoping Hu, Ting Liu, and Bing Qin. Question tells you where the answer is: Intention-aware long-context KV cache compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27153–27169, 2026. doi: 10.18653/v1/2026.acl-long.1250. ACL Anthology 2026.acl-long.1250.

A EVERY CLAIM, ITS EVIDENCE, AND ITS STATUS

Table 7 maps each claim the paper makes to the cell that carries it and to its status in the closure vocabulary of Appendix E. Reading guide: *interval-cleared* means the exact 95% interval clears the relevant bound; *confirmed* that it excludes zero; *gate met, wide* that the registered ± 3 pp point rule passed but $n{=}100$ cannot bound the interval; *ns* that no effect was detected at the stated power. Nothing in this table is claimed beyond its status column.

Table 7: Claim-by-claim evidence map. "Full pool" = the entire TyDiQA validation pool (669 Telugu / 113 Bengali). Registration: 19 of the 22 original arms were registered before any data existed; every later arm before its own first generation, after earlier results were known (Appendix R).

claim	evidence	status
default w=64 blinds bn/te; holes at scale	full pools, Table 9, App. L	interval-cleared (te $[-22.4, -17.3]$; bn -15, 3/20)
padding moves the English threshold with the pad	pad grid with own baselines, Fig. 2b	interval-cleared at all four pads: w=32 below c a loss, first window past c at baseline; blind 56–104 ns; collapse in slack LRT p=.13, post hoc
an English JSON schema blinds identically	Table 3B	interval-cleared close after resizing (3 tails non-inferior)
the threshold moves with the tokenizer	Gemma sweep (global layers only), App. I	consistent sweep on Bengali, n=100; Telugu -7 blind at 12B
$\hat{w}$ recovers the holes vs the default	full pools, App. E	interval-cleared ($+14.5$, $p=7\times10^{-18}$; $+13.3$, p=.0015)
closure gate at $\hat{w}$, five languages, n=100	Table 11	met on all five; non-inferior en+th; others wide
Q_{90} slack beats c+16 at scale	slack ablation, Section 6	interval-cleared ($+4.6$ $[+1.9, +7.0]$); vs c+32 ns; vs $c+Q_{50}$ $+1.9$ $[+0.1, +3.5]$
Telugu residual at $\hat{w}$, full pool	App. E	interval-cleared miss (-5.7 $[-7.8, -3.1]$)
the residual is not a visibility failure	IF + oracle full pools	confirmed (-3.4; oracle -5.1, head-to-head $+0.6$ ns)
the integer transfers to PyramidKV	App. O	significant ($+26$/$+31$, n=100)
shipped-constant flip costs th/sw	Section 4, n=200	significant (p=.022/.001)
an English instruction removes the damage	App. N	ns at n=100 (registered branch)
query-agnostic scorers lose everywhere	App. O	significant, English included
the per-item oracle window beats $\hat{w}$	oracle arm, App. E	refuted at n=669 ($+0.6$, $[-1.2, +2.3]$)
the hole and the remedy across ratios	Table 6, App. O	default monotone in r, ns at 0.5; $\hat{w}$ ties 256 on bn/te at every ratio (three predictions missed, reported)
the same integer works at twice the prefill	16k slice, App. M	interval-cleared hole; gate met, recovery p=.0023
the hole persists at 32B	App. M	interval-cleared: bn -12; te -10.2 at the full pool, its gate met at $\hat{w}$; bn misses (-5)
the shipped refine layout is damaged	App. T	ns (en -3; bn -8, p=.057) — geometry only
blind-but-undamaged is answerability	no-question control, App. I	refuted (bn keeps 68 of 83 vs en 77 of 97; te 35 of 75)
the blind window evicts the question	retention arm, App. J	refuted: 0.97–0.98 of the question's keys kept at w=64
the window buys gold-passage retention	retention arm, App. J	confirmed ($+0.074$ $[+0.059, +0.090]$; $+0.110$ $[+0.094, +0.127]$); blind ranking gold-poor, not gold-blind (second clause missed)
the fixed constant fails past its length	Telugu c=270, App. O	interval-cleared (-20 $[-23.5, -11.0]$; $\hat{w}$ $+14$ head-to-head)
repeating the question removes the hole	`qdup`, App. O	like question-last (te -7, bn -6; residual remains)

Table 7, continued

claim	evidence	status
the window buys no cache	App. B	measured: same entries and memory; scorer 15% of prefill at $\hat{w}$
the blind loss survives matched passage length	256-token chunks, App. J	bn interval-cleared (-17 $[-17.0, -10.4]$); te -11 $[-19.0, -0.3]$, middle branch, baseline cut by chunking
a public suite's own prompt blinds 24/26 languages; a retained needle costs nothing	OneRuler, App. T	0/0 in every cell; needle kept 0.58–0.63 blind (registered: no loss where kept)
the window's gain concentrates at the answer span	answer-span arm, App. J	refuted: answer kept 0.49–0.53 blind, above its passage; gain uniform ($+0.074$ vs $+0.074$); correctness follows the answer share (z=4.0 vs 2.7)
a public English template's tail blinds the shipped default	SWE-agent tail, App. T	void by its registered floor (baseline 81 $<$ 83); measured -41 $[-45.7, -30.5]$ at w=64, $+35$ back at $\hat{w}$
the threshold on the second tokenizer with a working Telugu baseline	Gemma-3-12b-it (global layers only), App. I	bn in full (blind -10 $[-11.9, -2.8]$; $\hat{w}$ $+8$ over it); te blind -7 $[-8.9, -0.3]$ at w=16, visible -5 (0/5), gate missed — middle branch
the headline cells on a second GPU family	`xstack_qwen`, App. R	within two points (-21/-2, -17/-3); 59–85% byte-identical
the loss follows the blind ranking's gold share across layouts and tokenizers	retention arm, App. J	monotone over 19 cells (Spearman $+0.76$, p=.0002); no cross-model threshold; question-first gold$-$distractor $+0.04$/$+0.06$ (registered $2\times$ missed)
the library's context-then-question path is a safe alternative	`ctxq`, App. O	refuted: interval-cleared loss in all eight languages (en -29 $[-33.7, -18.9]$; -23 to -42); the two-phase twin within ± 2 of baseline

B How last-w scoring works

SnapKV forms $q_{\text{obs}} = q[:, :, -w :]$, scores the prefix keys against that slice, keeps the highest-scoring entries under the compression ratio, and protects the window itself from eviction. The kvpress implementation defaults to w=64 with a pooling kernel of size 5, and the RocketKV path skips sparse KV entirely when the sequence is shorter than its prompt budget. kvpress's text-generation pipeline takes a context and a question separately and compresses the context alone; the runs here apply the press over the full prompt (`runner.py`), the setting in which the window is meant to reach the question, and the layout of every instruction-last item; the pipeline's own path is measured in Appendix O (Table 27). Because the window is retained rather than added, raising w at a fixed compression ratio does not increase the amount of cache that survives: it changes which entries are selected, not how many. A preregistered measurement (Table 8; 50 Telugu prompts, 8k prefill, r=0.75, one A6000, one warm-up, `torch.cuda` timing with synchronisation) confirms the first half and prices the second: the retained entries per layer and the peak memory are identical at w=64, 247 and 256, decode speed is unchanged, and the scorer's own time grows from 0.07 s to 0.18 s — 6.5% to 15% of prefill, $+10$% end to end. The registered expectation was a scorer share of about 5% or less; it is not, and the main text says so.

Table 8: Efficiency at a fixed ratio, r=0.75, Qwen3-4B, 50 Telugu prompts of 8k tokens; means over prompts. Scorer: time inside the press's `compress`. Kept: retained keys per layer. Prefill ranges span at most 0.03 s; the decode rates overlap across windows (noise at n=50).

	prefill (s)	scorer (s)	decode (tok/s, min–max)	peak (GB)	kept / layer
uncompressed	1.01	—	33.9 (32.8–35.4)	11.85	8,165
w=64	1.07	0.07	31.7 (31.0–33.9)	10.94	2,041
$\hat{w}$=247	1.18	0.18	32.4 (31.3–34.5)	10.94	2,041
w=256	1.18	0.19	32.8 (29.8–33.9)	10.94	2,041

C MEASURING c AND Q_{90}

c is produced by `measure_c.py`. A probe prompt is wrapped in the model's chat template with `add_generation_prompt`; if the probe question's token ids appear as a contiguous subsequence, c is the number of tokens after that span (`after_question`); otherwise $c = I + s$, where s is the assistant-header suffix measured on a minimal turn (`fallback_I_plus_suffix`). Qwen3-4B takes the fallback path with s=5 and Gemma-3-4b-it the direct path with s=6. We verified the fallback against a direct offset-mapping measurement on real prompts: for all eight Qwen languages the two agree within one token, the exception being a single token of Thai where the boundary falls inside a merged token.

The instruction's second sentence exists only to make scoring deterministic, so `measure_c.py --no-marker` also reports c without it, cutting at the last sentence-ending punctuation before the marker: 14/16/19/21 for English, Chinese, Spanish and Vietnamese, 26 for Swahili, 69 for Bengali and 98 for Telugu. Thai joins its two clauses with a conjunction and no punctuation, so it has no separable format sentence and we do not cut it on a guess. The headline cells do not rest on that clause: a minimal one-sentence instruction still leaves Bengali at 69 and Telugu at 98, past the default window, and under minimal instructions the blind set at w=32 still holds Bengali and Telugu — and Thai, whose instruction has no separable sentence to cut, remains blind there under its full c=45 either way.

For a padded or schema-carrying instruction, `measure_c_schema.py` builds the instruction with the same padding routine the item builder uses and reports c for the padded string, so the measured constant and the generated prompt cannot drift apart.

`measure_q.py` produces Q_{90} as the 90th percentile of question token length, `len(tokenizer.encode(question))`, on a split that excludes the 100 evaluation items: XQuAD validation after index 100, and TyDiQA-GoldP train with every evaluation question id removed. The evaluation items themselves are never used to estimate it.

Two TyDiQA-GoldP raw-split warts are disclosed rather than silently patched. Bengali's raw train and validation splits share three exact duplicate examples; all three sit inside `validation[:100]` and are therefore removed from the Q_{90} source by the evaluation-id exclusion above. Swahili's splits share two, at validation positions 133 and 300 — outside the n=100 evaluation set, so they remain in the estimation pool; recomputing $Q_{90,\mathrm{sw}}$ with them removed leaves the percentile unchanged (20, n=2753 vs 2755). Telugu's splits share no question text at all (0 of 5,563 train rows against the 669 validation questions), and $Q_{90,\mathrm{te}}$ is 80 whether the 100 evaluation items or all 669 are excluded. In every arm the invariant actually relied on holds and is asserted by the driver before generating: no evaluated item contributes to any Q_{90} source (for Swahili the evaluated set is `validation[:100]`). The guard that found the Swahili pair aborted a pod run before any generation existed; the log and the registered amendment ship with the record.

D THE SCORING RULE AND ITS ROBUSTNESS

An answer counts as correct if, after NFKC normalization, lowercasing, punctuation stripping and language-aware tokenization (character-level for scripts written without word separators; article stripping only for languages that have articles), some gold span appears inside the marked answer span or inside the first sentence of the prose that precedes the marker. Whole-output containment would over-credit answers that merely quote a passage; marker-only scoring under-credits models

that state the answer in prose and then emit a reformatted token after the marker, and that failure rate differs by language, which would manufacture exactly the kind of per-language gap this paper is about. An earlier frozen rule recorded during generation was marker-only; we never quote it and recompute every cell offline.

The choice does not carry the results. Under a stricter marker-only rule the Telugu dose ladder still shows the same shape: the uncompressed baseline scores 50% rather than 56%, $c{+}4$ scores 35% and $c{+}Q_{90}$ scores 47%, so the hole is -15 pp instead of -18 pp and the $c{+}Q_{90}$ cell is -3 pp instead of 0 pp. Applied to the full-pool Telugu re-run of Appendix E, the same rule leaves the interval-cleared residual in place: -9.4 pp (25 fixed / 88 broken, $p{=}2.1{\times}10^{-9}$, interval $[-11.9, -6.5]$). Table 10 repeats the 8B Bengali cells under two further scorers. A span match can miss a correct answer given in another inflected form, which matters most for Bengali and Telugu; the headline cells were therefore re-scored under a looser rule that credits a contiguous common substring covering at least 70% of the gold (an inflected form sharing its stem counts; scattered letter overlap does not). Baselines rise — Telugu 62.6 to 76.1, Bengali 71.7 to 81.4 on the full pools, English unchanged at 93 — and every ordering and every interval-cleared cell survives: Telugu -21.5 at the default ($[-23.9, -18.4]$) and -7.5 at $\hat{w}$ ($[-9.4, -5.0]$); Bengali -13.3 ($[-17.5, -5.1]$) and -1.8 ($[-4.9, +2.9]$).

Table 9 re-scores the headline cells under three stricter rules at once. The Telugu hole survives each of them — -22 at the default under marker-only containment, -8 under strict exact match on the marker span, -16.5 under token F1 — and so does $\hat{w}$'s full-pool residual (-9.4, -4.9, -6.8). Bengali states its answer in the sentence before the marker and echoes the template's placeholder after it (96 of 100 baseline outputs carry a marker; marker-only containment credits 3 and exact match 2), so those rules score format there, and the strict rule that applies is F1 on the answer span, which gives -7.2 at the default and -3.0 at $\hat{w}$ (-7.4 and -2.7 on the full pool). No rule changes a sign or an ordering; the lenient rule is the one that scores every language on the same footing, which is why it carries the headline.

Table 9: The headline cells under four scoring rules: accuracy and paired Δpp against the cell's own baseline. Lenient: containment on the marker span or the first sentence (the paper's rule). Marker: containment on the marker span only. EM: strict exact match on the marker span. F1: token F1 on the marker span (mean). $n{=}100$; full pools 669 (te) and 113 (bn).

	cell	lenient	marker	EM	F1
en	baseline	93	76	68	76
	$w{=}64$	93 (0)	74 (-2)	66 (-2)	74 (-1.8)
	$\hat{w}{=}43$	95 ($+2$)	75 (-1)	67 (-1)	74 (-1.6)
bn	baseline	73	3	2	33
	$w{=}64$	57 (-16)	1 (-2)	2 (0)	25 (-7.2)
	$\hat{w}{=}183$	71 (-2)	1 (-2)	1 (-1)	30 (-3.0)
te	baseline	56	50	28	49
	$w{=}64$	37 (-19)	28 (-22)	20 (-8)	33 (-16.5)
	$\hat{w}{=}247$	56 (0)	47 (-3)	27 (-1)	47 (-2.6)
te, 669	baseline	63	55	36	56
	$w{=}64$	42 (-20.2)	35 (-19.6)	24 (-11.8)	39 (-17.0)
	$\hat{w}{=}247$	57 (-5.7)	46 (-9.4)	31 (-4.9)	49 (-6.8)
bn, 113	baseline	72	4	3	33
	$w{=}64$	57 (-15.0)	2 (-1.8)	2 (-0.9)	25 (-7.4)
	$\hat{w}{=}183$	70 (-1.8)	2 (-1.8)	2 (-0.9)	30 (-2.7)

Table 2 reports 25 paired tests. Under a Holm correction across the grid, five blind cells survive — Bengali at $w{=}56$ and 88, Telugu at $w{=}32$, 56 and 88 — and the single-window Thai and Swahili dips, Bengali at $w{=}32$ ($p{=}.0072$, Holm threshold .0025), and the Telugu $w{=}120$/176 cells do not. No claim in the paper rests on an unsurviving cell alone: the Telugu $c{+}\varepsilon$ point is carried by the dose ladder, whose $c{+}4$ cell stands at $p{=}.0009$ in the five-rung ladder's own family (Holm threshold .01).

Table 10: Robustness of the 8B Bengali cells to the scoring rule. The spelling fold maps two pairs of Bengali graphemes that the model interchanges; gold-token recall accepts a prediction covering at least 70% of the gold tokens. The ranking and the sign of every comparison are unchanged.

scorer	baseline	w=64	$\hat{w}$	$\hat{w}-$ baseline
containment (used throughout)	81	64	73	−8
containment after a spelling fold	81	65	75	−6
gold-token recall ≥ 0.7	85	70	79	−6

E CLOSURE, WITH INTERVAL ESTIMATES

The registered closure rule — $|\Delta| \leq 3$ pp against the uncompressed baseline, as a point estimate — is a decision rule fixed before each run, not an equivalence test, and at n=100 the two are far apart. Table 11 reports an exact 95% interval for every closure cell, obtained by conditioning on the discordant count N: given N, the number of items the treatment fixes is binomial, and a Clopper–Pearson interval for its rate maps monotonically onto an interval for Δ. Six cells support the stronger reading that the lower bound excludes a loss beyond −3 pp: English at 4B and at Llama, Thai at its own $\hat{w}$, and all three schema tails. The Bengali and Telugu intervals at 4B are consistent with the gate they met but also with losses the gate would have rejected — which is what n=100 can and cannot say — and the same holds for the Gemma, Llama and instruction-first cells. The 8B interval excludes zero, confirming that residual. The recovery claims against the default window (+9 to +19 pp, $p\leq$.022, on 4B, 8B and Llama; 32B +7 ns and +8.5 interval-cleared) are unaffected: they are significance claims and carry their own tests.

Two of those cells were then re-run at depth. The stopping rule was fixed before any of it existed: final n is the entire validation pool, 669 Telugu and 113 Bengali items, a rule with no free parameter and therefore nothing to stop early on. Telugu's interval now lies entirely below the gate rather than merely below zero — the stronger reading the table calls an interval-cleared residual — and the decomposition is exact: the original hundred items reproduce byte-identically and still sit at 0.0, while the 569 new ones sit at −6.7. Telugu's 57 residual items are no poorer in fully visible questions than the pool: 51 of 57 (89%) have $|Q| \leq Q_{90}$=80 against 611 of 669 (91%) in the pool (Fisher p=.62; $|Q|$ recomputed from the tokenizer by the audit) — the visibility diagnosis Section 6 quotes, in test form. Bengali's thirteen items beyond the original hundred contain no discordant pair at all: its full-pool row is the n=100 row over a larger denominator, not an independent confirmation — the pool is simply exhausted. Bengali meets the gate on a pool its own size caps at 113, so its interval stays wide, as the preregistration said it would. The 8B Bengali cell is consequently no longer the only residual in this paper that an interval confirms; the primary model now carries one too, measured at 6.7× the sample.

Three later preregistered arms extend the table. The slack ablation re-ran the full Telugu pool at c+16, c+32 and $\hat{w}$ in one self-contained store (baselines byte-identical to the depth arm's on all 1,338 shared generations): each fixed slack leaves an interval-cleared residual strictly deeper than $\hat{w}$'s, and the registered head-to-head gives $\hat{w}$ +4.6 pp over c+16 (55/24, p=.0006, $[+1.9, +7.0]$) and +2.1 over c+32 ($[-0.3, +4.2]$, ns). Under the marker-only scorer the ordering repeats (43.9/45.1/45.6 against a 55.0 baseline) with the c+16 head-to-head at +1.6 ($[-1.1, +4.3]$), directionally consistent and individually not significant. A later preregistered run put the median question beside the percentile on the same pool (store `slack_q50`; its baseline and $\hat{w}$ rows repeat the slack ablation's 1,338 byte for byte): $c+Q_{50}$=221 leaves −7.6 ($[-9.6, -5.1]$), and the head-to-head favours $\hat{w}$ by +1.9 (24/11, p=.041, $[+0.1, +3.5]$). The interval clears zero and the point falls short of the 3 pp the registration asked for, so neither registered branch fires and the number is reported as measured; the marker-only scorer gives +1.6 in the same direction. The instruction-first arm re-ran the same pool with the question last: the confirmed −3.4 residual in the table is what eviction costs when visibility cannot be the cause, and its broken items show no concentration by gold position or question length (Fisher $p \geq .17$; marker-only scoring is uninformative in this layout, which suppresses the marker itself — the Appendix S behaviour). The Thai and Swahili rows are the closure gate at the two languages the rule had never been run on; both meet it, Thai with a non-inferior interval, Swahili at the point boundary with the interval wide — so the rule's evaluated set is five of the eight languages, and no new miss joins the residual table.

Table 11: Exact-conditional 95% intervals for the paired difference at $\hat{w}$; `closure_cis.py`. The instruction-first rows are mechanism-gate cells — the default window with the question last, so $V{=}1$ by construction — judged by the same rule.

cell	n	Δpp	fixed/broken	95% CI	reading
4B en	100	$+2$	2/0	$[-1.4, +2.0]$	non-inferior at -3 pp
4B bn	100	-2	6/8	$[-9.1, +5.9]$	interval too wide
4B te	100	0	6/6	$[-6.9, +6.9]$	interval too wide
4B th	100	-1	1/2	$[-2.9, +2.4]$	non-inferior at -3 pp
4B sw	100	-3	2/5	$[-6.5, +2.9]$	gate met; interval wide
full pool: te	669	-5.7	19/57	$[-7.8, -3.1]$	interval-cleared residual
full pool: bn	113	-1.8	6/8	$[-8.0, +5.2]$	gate met; interval wide
full pool: te $c{+}16$	669	-10.3	24/93	$[-12.7, -7.4]$	interval-cleared residual
full pool: te $c{+}32$	669	-7.8	18/70	$[-9.8, -5.2]$	interval-cleared residual
full pool: te $c{+}Q_{50}$	669	-7.6	17/68	$[-9.6, -5.1]$	interval-cleared residual
full pool: te IF at $w{=}64$	669	-3.4	16/39	$[-5.3, -1.2]$	confirmed residual
full pool: te oracle w_i	669	-5.1	22/56	$[-7.3, -2.4]$	confirmed residual
te @16k, $w{=}64$	100	-15	3/18	$[-19.7, -5.7]$	interval-cleared residual
te @16k, $\hat{w}$	100	-2	3/5	$[-6.6, +4.1]$	gate met; interval wide
32B bn	100	-5	2/7	$[-8.5, +1.8]$	gate missed; interval wide
32B te, full pool	669	-1.6	16/27	$[-3.5, +0.4]$	gate met; interval wide
refine en at $w{=}64$	100	-3	1/4	$[-4.9, +2.2]$	interval too wide
refine bn at $w{=}64$	100	-8	3/11	$[-12.7, +0.2]$	interval too wide
JSON-60	100	$+2$	3/1	$[-2.4, +3.9]$	non-inferior at -3 pp
JSON-120	100	$+1$	1/0	$[-0.9, +1.0]$	non-inferior at -3 pp
JSON-200	100	$+2$	2/0	$[-1.4, +2.0]$	non-inferior at -3 pp
Gemma bn	100	-6	3/9	$[-10.7, +1.7]$	interval too wide
Gemma te	100	-3	4/7	$[-8.6, +4.2]$	interval too wide
8B bn	100	-8	0/8	$[-8.0, -2.1]$	confirmed residual
Llama en	100	$+1$	1/0	$[-0.9, +1.0]$	non-inferior at -3 pp
Llama bn	100	-8	3/11	$[-12.7, +0.2]$	interval too wide
Llama te	100	-1	3/4	$[-5.6, +4.4]$	interval too wide
IF bn at $w{=}64$	100	-4	3/7	$[-8.7, +3.0]$	interval too wide
IF te at $w{=}64$	100	-6	1/7	$[-7.9, +0.4]$	interval too wide
chunked te, $\hat{w}$	100	-6	4/10	$[-11.7, +2.3]$	interval too wide
chunked bn, $\hat{w}$	100	-9	2/11	$[-12.5, -1.2]$	confirmed residual

F FULL WINDOW, PAD, AND $c{+}16$ GRIDS

Table 12 gives the measured constants for all eight languages. Its "passages" row is the mean number of passages packed into the 8k prefill: fertility sets the trailing block and the prefill composition together, so a Telugu or Bengali item holds about 14 passages where an English one holds 39 — a confound for comparing the depth of damage across languages, not for any within-language cell. Table 13 gives the eight-language run of the $c{+}16$ ablation: the five languages whose trailing block already fits inside the default window stay within 3 pp of baseline at both settings, with a marginal Vietnamese exception at the default window; only Bengali and Telugu carry a hole for $c{+}16$ to close, and it does not close it. Table 14 gives the English pad sweep plotted in Figure 2b, from a preregistered re-run that put an uncompressed baseline beside each pad (`pad_base`; its 2,000 compressed generations are byte-identical to the original sweep's). The collapse test of Section 4 was registered to be re-run on that store three ways, and all three are reported: on the compressed cells (pad fixed effects, likelihood ratio $p{=}.13$; pad$\times$slack $p{=}.82$), with the baseline as a sixth level per pad ($p{=}.17$ and $.78$), and on the paired per-item difference against the pad's own baseline, where pad identity is detectable at $p{=}.029$ (interactions $p{=}.39$). The grid therefore collapses on the accuracy scale and not cleanly on the paired scale; every variant is post hoc, ignores item clustering (each item appears five times per pad), and the paired outcome is three-valued and fitted by least squares, so we quote the compressed-cell test in the main text and none of the three as established.

The Vietnamese exception is worth pinning down, because Vietnamese is the one safe-set language whose cell brushes the gate (-5 at w=64). Its held-out question percentiles on this tokenizer are $Q_{50}/Q_{90}/Q_{\max} = 17/26/46$ (n=1090), and with c=39 the default window leaves 25 tokens of slack — so only 3 of the 100 evaluation items have $V < 1$ at all. The five items the default window breaks all have V=1: none of the damage sits on the partially visible items, so the -5 is ordinary compression residual of the kind Section 6 diagnoses, not a shallow copy of the Bengali–Telugu blind mode. The rule would set $\hat{w}_{\text{vi}} = 65$, one token above the shipped default.

Table 12: Measured constants, Qwen3-4B. I is the instruction length, c=I+s with s=5, and Q_{90} is the held-out question-length percentile (n=1090/2387/5563 for en/bn/te; Q_{50} is 12/46/54 and $Q_{\max}$ is 36/141/170). Q_{90} was measured for the languages the AutoWindow arms use, for the two languages the w=32 flip touches in the sweep, and for Vietnamese (Appendix F); Spanish also has $c > 32$ but is not swept. The character and whitespace-word counts of the same instructions (words omitted for the two scripts without word spaces) show that the token spread is the tokenizer's, not the translation's: Telugu is 1.15× English in characters and 8.1× in tokens; the sub-character row counts pieces that do not decode to a whole character (byte fallback), of which Gemma-3's vocabulary has none.

	en	zh	es	vi	th	sw	bn	te
I	20	24	30	34	40	42	102	162
I, characters	98	41	126	112	93	104	102	113
I, words	16	—	19	26	—	17	14	14
I, sub-character tokens	0%	0%	0%	0%	0%	0%	59%	81%
I on Gemma-3	21	25	29	34	32	36	26	38
passages per 8k prefill	39	50	46	33	20	65	15	14
c	25	29	35	39	45	47	107	167
c+16	41	45	51	55	61	63	123	183
Q_{90}	18	—	—	26	46	20	76	80
$\hat{w} = c$+Q_{90}	43	—	—	65	91	67	183	247

Table 13: The c+16 ablation across eight languages. Accuracy, with paired Δpp against the uncompressed baseline in parentheses; n=100; star: McNemar $p < .05$. Shading marks the blind cells, as in Table 2: of the eight languages, the shipped window blinds two.

	c	c+16	baseline	w=64	c+16
en	25	41	93	93 (+0)	94 (+1)
zh	29	45	83	83 (+0)	85 (+2)
es	35	51	88	86 (−2)	88 (+0)
vi	39	55	83	78 (−5)	81 (−2)
th	45	61	85	85 (+0)	84 (−1)
sw	47	63	56	53 (−3)	56 (+0)
bn	107	123	73	57 (−16*)	66 (−7)
te	167	183	56	37 (−19*)	50 (−6)

Table 14: English pad sweep. Accuracy with paired Δpp against each pad's own uncompressed baseline, n=100 per cell, eviction ratio 0.75; star: McNemar $p < .05$, which here coincides with the exact interval excluding zero. Shading marks the blind cells ($w < c$); the bold cell in each row is the first window past c.

pad	c	baseline	w=32	56	80	104	144
48	57	92	81 (−11*)	87 (−5)	**92** (+0)	92 (+0)	92 (+0)
64	73	94	86 (−8*)	90 (−4)	**94** (+0)	93 (−1)	93 (−1)
96	105	94	82 (−12*)	90 (−4)	89 (−5)	90 (−4)	**93** (−1)
128	137	91	78 (−13*)	85 (−6)	89 (−2)	89 (−2)	**91** (+0)

G THE DOSE LADDER IN FULL

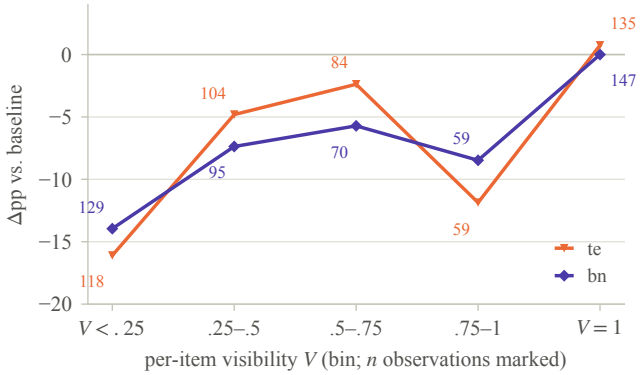


Figure 3: Paired damage by per-item visibility V, pooling the five treated rungs of each ladder (Table 15); the bin sizes count observations. Both ladders lose double digits below $V{=}0.25$, recover partially through the middle bins, break in the $0.75 \le V < 1$ band on long-question items, and show no detectable effect at $V{=}1$.

Table 15: The dose ladder. $n{=}100$ per rung, ratio 0.75, paired against each language's own baseline; median V is the per-item share of the question inside the window. The last row of each block is that language's $\hat{w} = c{+}Q_{90}$.

w	slack $w{-}c$	median V	accuracy	Δpp	fixed/broken	p
Telugu, $c{=}167$, baseline 56						
171	4	0.08	38	-18^{*}	5/23	.0009
183	16	0.31	50	-6	5/11	.21
199	32	0.62	52	-4	4/8	.39
215	48	0.92	52	-4	6/10	.45
247	80	1.00	56	0	6/6	1.0
Bengali, $c{=}107$, baseline 73						
111	4	0.09	56	-17^{*}	3/20	.0005
123	16	0.34	66	-7	5/12	.14
139	32	0.68	70	-3	4/7	.55
155	48	1.00	68	-5	3/8	.23
183	76	1.00	71	-2	6/8	.79

The Telugu ladder is the upper block of Table 15. Binning its five treated cells by per-item visibility gives: $V < 0.25$, $n{=}118$, -16.1 pp (6 fixed / 25 broken, $p{=}9 \times 10^{-4}$); $0.25 \le V < 0.5$, $n{=}104$, -4.8 pp; $0.5 \le V < 0.75$, $n{=}84$, -2.4 pp; $0.75 \le V < 1$, $n{=}59$, -11.9 pp (2/9, $p{=}.065$); $V{=}1$, $n{=}135$, $+0.7$ pp (9/8). Because an item can revisit a bin at more than one rung, the n here count observations, not items: the $V{<}0.25$ bin holds 118 observations of 100 distinct items and the $V{=}1$ bin 135 of 93. Recomputed with each distinct item counted once (its mean treated correctness in the bin against its baseline), the two ends move to -18.0 and 0.0 pp — the multiplicity flattered the blind end slightly and the conclusion is unchanged. On the treated rows AIC prefers $y \sim V$ (689.8) to $y \sim (w{-}c)$ (692.1) and to $y \sim (w{-}c) + |Q|$ (692.2) — a gap of two is support, not a discrimination.

The $0.75 \le V < 1$ band is the one that breaks monotonicity, so we describe its contents. Its 59 observations are 59 distinct items — each item enters the band at exactly one rung, since visibility rises with the window — and they arrive through three of the five: 14 at $c{+}32$, 40 at $c{+}48$ and 5 at $c{+}80$. The damage is carried by nine of them, with questions of 40 to 81 tokens; six of those nine break at $c{+}48$, and four are answered correctly once the window reaches $V{=}1$. The band is not significant on its own ($p{=}.065$) and it represents 59 of the 500 treated observations. Read together with the $V{=}1$ bin, it says that visibility close to 1 is not yet equivalent to visibility at 1 for long

questions — which is the same statement as the registered leftover of a language-level percentile, seen per item.

The Bengali ladder, the lower block, was run on its own grid with the same readout script and repeats the shape. Its bins are $V < 0.25$, n=129, -14.0 pp (5 fixed / 23 broken, p=9×10^{-4}); $0.25 \le V < 0.5$, n=95, -7.4 pp; $0.5 \le V < 0.75$, n=70, -5.7 pp; $0.75 \le V < 1$, n=59, -8.5 pp (2/7, ns); V=1, n=147, no detectable effect (0.0 pp, 9/9). The same per-item dedup applied here moves the two ends to -15.0 (129 observations, 100 items) and -0.4 pp (147 observations, 88 items). Both ends land where the account says they should, and the band that breaks monotonicity is the same one, which makes the caveat above a two-language observation rather than an artefact of one grid. Across the ladder the cluster-robust logit on V gives $+0.65$ (z=2.95, p=.003); within items, 18 switchers are correct at high visibility against 1 in the opposite direction (p=10^{-4}), and AIC prefers V (637.7) to $(w{-}c)$ (639.7) and to $(w{-}c) + |Q|$ (641.5) — again a gap of two, which is support rather than a discrimination. An earlier run of this block was cut short at 43 items on its third rung and tied on that last comparison; the tie resolves, narrowly, once the ladder is complete. Its numbers are superseded here, and the 343 generations the two runs share are byte-identical.

One-parameter descriptions of the sweep. We also compared one-parameter descriptions of the compressed rows, and the comparison does not fully favour us. AIC prefers language fixed effects (2744) over the distance to the threshold $(w{-}c)$ (3046), the binary blindness indicator (3075), and the raw window w (3220). The mechanism's prediction that $(w{-}c)$ beats raw w holds. But on a five-language grid $(w{-}c)$ is nearly collinear with language identity, and the sweep alone therefore cannot separate a threshold from a per-language constant.

H SCHEMA TAILS: THE MIS-SIZED RUN AND THE RERUN

The first schema run sized its window from the *unpadded* English instruction with the 16-token slack of the $c{+}16$ ablation ($25{+}16{=}41$), not from the schema-adjusted c, giving w=41 — smaller than the default w=64 it was supposed to improve on. It therefore tested a window below the threshold rather than the rule, and the result was correspondingly bad: against uncompressed baselines of $94/95/94$ for the 60-, 120- and 200-token schema tails, the default window scored $88/89/89$ (-6, -6^*, -5 pp) and the mis-sized window scored $88/72/75$ (-6, -23^*, -19^* pp). The mis-sizing is itself an instance of the paper's claim: an English-derived constant applied to a prompt whose trailing block is much longer.

The rerun measures c with the schema in place ($72/166/213$) and is reported in panel (B) of Table 3. The two runs share their uncompressed and default-window cells, which the rerun reproduced exactly, generation for generation.

I A SECOND TOKENIZER: GEMMA-3-4B-IT AND GEMMA-3-12B-IT

Gemma-3-4b-it encodes the same instructions much more compactly than Qwen3-4B: c is $27/32/44$ for English, Bengali and Telugu against Qwen's $25/107/167$, and its questions are shorter in tokens as well (Q_{90} = $18/18/20$ against $18/76/80$). The threshold should therefore move down with the tokenizer, and the window sweep in Table 16 is consistent with that: Bengali is damaged at the two windows below its c and drifts to a non-significant -5 pp at the two above it, while Telugu — whose Qwen counterpart loses 19 points at w=64 — shows no comparable hole anywhere on the grid. English shows no detected damage anywhere on the grid.

Table 16: Gemma-3-4b-it window sweep. Paired Δpp against the Gemma baseline (accuracy $75/62/37$ for en/bn/te), n=100, star: McNemar $p < .05$. Shading marks blind cells as in Table 2. Not pooled with any Qwen cell.

	c	w=16	24	32	48	64
en	27	+2	+1	+3	+4	+3
bn	32	−13*	−10*	−9*	−5	−5
te	44	−3	−4	−3	−1	−3

Three qualifications belong with this table. Bengali at w=32, which is c exactly — still total blindness by the V formula, since the window holds only the trailing block — loses 9 pp, consistent with its blind neighbours; the registered shading rule $w < c$ leaves that boundary cell unshaded, a one-token conservatism we note rather than repaint; Telugu's baseline of 37% is low enough that the sweep would not resolve a small hole (the 12B run below, with a baseline of 61, resolves one); and English is itself blind at w=16 and 24 on this tokenizer without being damaged there — indeed no window we tested damages Gemma's English at this ratio. Blindness is necessary for the failure but not sufficient. Part of the reason is the instrument: kvpress compresses only Gemma 3's global layers (5 of 34 here, 8 of 48 at 12B), so a blind window on Gemma is blind in those layers alone while every sliding-window layer keeps the question inside its 1,024-token window (Appendix J). How much any ranking is worth is measured in Appendix O: evicting at the same ratio with no ranking at all collapses generation itself, so even a blinded ranking does most of the work. What distinguishes Gemma's undamaged blind English cells from Qwen's damaged padded ones, that measurement cannot say. A preregistered answerability control (gold passage only, no distractors, the question removed) tests the cheapest account — that the undamaged cells hold items answerable without their question — and its separating clause is refuted: English keeps 77 of its 97 gold-only points without the question and Bengali 68 of 83, so answerability does not separate the language the blind cells damage from the one they spare; Telugu scores 35 without the question against 75 with it (gold-only), with 87% of its no-question outputs running to the decode cap as passage summaries, which the first-sentence rule then scores. We report the asymmetry as an observation with one account excluded.

The same tokenizer at 12B. Whether Telugu's null was the floor was testable: Gemma-3-12b-it shares Gemma-3-4b-it's tokenizer and template (c and Q_{90} re-measured on the pod, identical), so a preregistered arm (`iclr-gemma12b-preregister.md`) re-ran the sweep at w=16, 32, 64 and $\hat{w}$ on the same items, own baselines (Table 17; `gemma12b`). With a Telugu baseline of 61 the blind w=16 loses -7 (1/8, $[-8.9, -0.3]$), one point short of the registered -8, so the 4B null was at least in part the floor; but the visible w=64 sits at -5 (0/5, $[-5.0, +0.2]$), misses the gate, and recovers only $+2$ over the blind window (ns): on this tokenizer Telugu's loss is mostly the residual a visible window does not remove, as on Qwen's full pool, and the blind mode adds about two points on top. Bengali reproduces the whole shape: blind -10 and -9 at w=16 and 32 (both intervals below zero), -5 at w=64, -2 at $\hat{w}$=50, a $+8$ recovery over the blind window ($[+0.4, +11.5]$). English is flat at every window, blind or not. So the second tokenizer reproduces the threshold on Bengali in full and on Telugu at the blind extreme only; "blindness is necessary, not sufficient" stands, with the Telugu cell a shallow hole rather than a null.

Table 17: Gemma-3-12b-it window sweep, n=100, ratio 0.75, own baselines; paired Δpp with fixed/broken counts and exact 95% intervals; star: McNemar $p < .05$. V=0 marks the blind cells (Bengali's w=32=c holds no token of the question).

	window	V	acc	Δpp	fixed/broken	95% CI
en, c=27, base 91	w=16	0	91	0	1/1	$[-1.9, +1.9]$
	w=32	partial	92	+1	1/0	$[-0.9, +1.0]$
	w=64	1	91	0	1/1	$[-1.9, +1.9]$
	$\hat{w}$=45	1	90	−1	0/1	$[-1.0, +1.0]$
bn, c=32, base 80	w=16	0	70	-10^*	1/11	$[-11.9, -2.8]$
	w=32	0	71	-9^*	2/11	$[-12.5, -1.2]$
	$\hat{w}$=50	1	78	−2	1/3	$[-3.9, +2.4]$
	w=64	1	75	−5	1/6	$[-6.9, +1.1]$
te, c=44, base 61	w=16	0	54	-7^*	1/8	$[-8.9, -0.3]$
	w=32	0	57	−4	4/8	$[-9.6, +3.6]$
	w=64=$\hat{w}$	1	56	−5	0/5	$[-5.0, +0.2]$

The same control bounds a second account: that the blind window does not merely hide the question from the scorer but lets the question's own keys be evicted, so the model answers without it. The output form says otherwise (Table 18): at w=64 the blind Telugu and Bengali outputs carry the answer marker and stop short of the decode cap at the baseline's rates, while the no-question control runs 87% of Telugu outputs to the cap with the marker on 24%. Whatever the default window

evicts, it leaves the question readable at decode; the accuracy it loses is in what is answered. The index-level version of this measurement — which keys the blind ranking keeps — is Appendix J. (Two early-stack observations belong here rather than in the main text: 4-bit KV-quantization pilots damaged languages in a non-fertility order, and at this ratio no Bengali damage gradient appeared across gold-length or retention strata, not an interval-cleared absence; neither is evidence about the window.)

Table 18: Output form at the default window against the no-question control. Marker: share of outputs carrying the answer marker; cap: share running to the 384-token decode cap. Telugu on the full pool (`depth`), Bengali and English at n=100 (`autowin`), the control from `noq`.

	cell	n	marker	cap
te	uncompressed	669	87%	9%
	w=64, blind	669	83%	11%
	$\hat{w}$=247	669	83%	10%
	gold only, question removed	100	24%	87%
bn	uncompressed / w=64	100	96% / 96%	6% / 6%
	gold only, question removed	100	39%	62%
en	uncompressed / w=64	100	99% / 99%	0% / 0%

The within-model contrast is untouched — the same two blind windows cost Bengali 13 and 10 points — but the shading in Table 16 makes the asymmetry visible, and we would rather state it than let a reader find it. The AutoWindow arm on this tokenizer gives $\hat{w} = 45/50/64$ and returns $+3/-6/-3$ pp against baseline, against $+3/-5/-3$ at the default window. For Telugu the rule selects the default constant itself. For Bengali it does not close, and the main text explains why we read that residual as a plateau rather than a threshold effect: it is present at every window from 48 upward, and at $\hat{w}$=50 the median Bengali question of 11 tokens is entirely inside the window — 7 of the 9 items it breaks are fully visible, against 86% of the pool (Fisher p=.61).

A stuffed arithmetic control. The task-generality check behind the Section 7 disconfirmation (store `stuffed`): MGSM grade-school problems with same-language distractor passages packed in front of them to 8k tokens, the problem placed immediately before the frozen native instruction, scored by numeric exact match rather than the span rule; English, Bengali and Thai, n=100, r=0.75, own baselines. Bengali's default window is blind by the same 107-token block and loses -3 pp (7 fixed / 10 broken, ns): the blind mode needs a question matched against passages, not merely critical text the window cannot see.

J What the blind ranking keeps

A preregistered arm (`iclr-mechanism-arms-preregister.md`, section B4) recorded the keys the ranking retains, per layer and head, with a subclass of the kvpress press that reproduces its `compress` step and logs the selected indices before the gather; a smoke test confirmed the subclass and the stock press generate byte-identical text on one item before the arm ran. Prefill only, no decode; 1,600 prefills over 36 layers. The prompt is built by the campaign's own builder, the question and gold-passage spans are recorded as character offsets and mapped to token positions with the tokenizer's offset mapping on the exact prompt string, and no item lost its spans. Per item, `q_keep` and `g_keep` are the shares of the question's and the gold passage's keys kept, averaged over heads then layers; `d_keep` is the same share for the mean distractor. Correctness is joined from the generation store of the same item and window.

Table 19: Share of a span's keys kept by the ranking, r=0.75, n=100 items per cell, means over items (median in parentheses for the gold passage); gold tok.: gold-passage tokens kept. Spans: Telugu question 55 tokens and gold passage 685; Bengali 53 / 687; English 13 / 147.

cell	V	question	gold passage	gold tok.	distr.	gold−distr.
te w=64, blind	0	0.97	0.41 (0.37)	241	0.32	+0.09
te $\hat{w}$=247	1	1.00	0.48 (0.39)	293	0.34	+0.15
bn w=64, blind	0	0.98	0.43 (0.38)	266	0.32	+0.11
bn $\hat{w}$=183	1	1.00	0.54 (0.45)	334	0.33	+0.21
en w=64	1	1.00	0.70 (0.67)	100	0.33	+0.38
en $\hat{w}$=43	1	1.00	0.71 (0.68)	102	0.32	+0.39
en pad 96, blind w=32–104	0	1.00	0.55 / 0.63 / 0.62 / 0.62	75–89	0.31	—
en pad 96, w=144	> 0	1.00	0.69	98	0.31	—
en pad 128, blind w=32–104	0	1.00	0.54 / 0.59 / 0.62 / 0.62	74–87	0.31	—
en pad 128, w=144	> 0	1.00	0.65	91	0.31	—

Five readings were registered. (1) The question survives the default window: `q_keep` at w=64 is 0.974 on Telugu and 0.983 on Bengali (tenth percentiles 0.951 and 0.971), against a registered threshold of 0.8; the account on which the blind window lets the question's own keys be evicted is refuted at the index level, in agreement with the output-form bound of Table 18. (2) What the window buys is the gold passage: paired over items, `g_keep` at $\hat{w}$ exceeds `g_keep` at w=64 by +0.074 on Telugu ([+0.059, +0.090] bootstrap, 83 of 100 items up, sign test p=10^{-11}) and +0.110 on Bengali ([+0.094, +0.127], 93/7, p=3×10^{-20}); on English, already at V=1, by +0.013. The reading's second clause missed: it predicted the blind ranking would keep the gold passage *below* a distractor's share, and it keeps it slightly above (Telugu +0.087, 60 of 100 items; Bengali +0.108, 56/100), rising to +0.147 and +0.213 at $\hat{w}$ — gold-poor, not gold-blind. (3) The graded English blind region is a retention effect: across the blind windows 32–104 of pad 128, `g_keep` rises 0.544 → 0.593 → 0.615 → 0.622 (Spearman +1; per item, w=104 over w=32 is +0.078, 99 of 100 up), and pad 96 rises 0.552 → 0.630 then plateaus (+0.069, 99/100), while `q_keep` is flat at 0.999–1.000; Table 14's accuracies follow the same shape. (4) Correctness follows retention within a cell: pooling the three campaign cells at w=64, a logit of correctness on `g_keep` with language fixed effects gives +1.82 (s.e. 0.66, z=2.74); the items answered correctly kept more of their gold in every language (0.44 vs 0.39 Telugu, 0.47 vs 0.37 Bengali, 0.71 vs 0.58 English). (5) The exploratory language contrast is confounded as registered: an English blind window keeps 0.54–0.63 of a 147-token gold passage where a Telugu one keeps 0.41 of a 685-token one, and this arm does not hold passage length fixed. Neither normalisation collapses the languages: Telugu at $\hat{w}$ keeps a smaller share than a blind English pad (0.48 vs 0.55) and three times the tokens (293 vs 75–89), and scores at baseline where the pad loses −12; so the mapping from retained gold to accuracy depends on passage length, and the depth of the blind loss across languages is not separated from it here.

Across layouts and tokenizers. Whether the loss follows what the blind ranking keeps, whatever put it there, was testable with the same hook. A preregistered arm (`iclr-retention-layouts-preregister.md`; its one amendment, made before any prefill, withdrew a threshold clause that the pad cells above already refuted) ran it on the question-first refine layout (Qwen3-4B, en/bn/te, w=64, blind by construction), on Gemma-3-4b-it (en/bn/te, w=16 blind and w=64) and on Gemma-3-12b-it (Telugu), and pooled every blind cell that has a paired damage measurement: 19 cells over three layouts, two tokenizers and three model sizes (Figure 4, Table 20; `retain_layouts`). The paired loss is monotone in the blind ranking's mean gold share (Spearman +0.76, permutation p=.0002): the damaged instruction-last cells sit at 0.41–0.43, and every blind cell with no detectable loss — the OneRuler needle, the question-first layout, Gemma's English, the English pads past w=56 — sits at or above 0.50. What does not exist is a threshold: Gemma-4B Bengali loses −13 at 0.59 where a Qwen English pad loses −2 at 0.62, and on Gemma the share is a share of five global layers — kvpress leaves Gemma 3's sliding-window layers uncompressed, and the question is inside every local layer's 1,024-token window. Question-first keeps a larger gold-minus-distractor contrast without the question in the window (+0.04 Telugu, +0.06 Bengali over instruction-last blind, paired per item; the registered 2× margin was missed), in the direction a key-side account predicts — passage keys computed after the question carry it — and it keeps far less of the question itself (0.70–0.82 against 0.97–1.00),

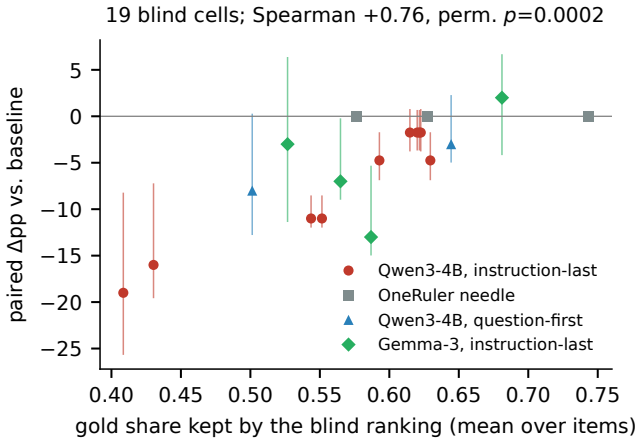


Figure 4: Paired loss against the gold share the blind ranking keeps, one point per blind cell (bars: exact 95% intervals): Qwen3-4B instruction-last (te, bn, and the English pads at w=32–104), OneRuler needles (hi, ta, sw), the question-first layout (en, bn) and Gemma-3 (4B en/bn/te and 12B te at w=16).

which sits where the ranking protects nothing. So the direction is measured and the number is not: the less gold a blind ranking keeps, the more is lost, on every layout and tokenizer we ran.

Table 20: The new cells of the pooled test: mean kept share of the gold passage, a distractor and the question, and the cell's paired loss (`retain_layouts`; Gemma's shares are over its global layers).

cell	gold	distr.	question	loss
refine en w=64 (question first)	0.64	0.32	0.82	-3 $[-4.9, +2.2]$
refine bn w=64	0.50	0.33	0.70	-8 $[-12.7, +0.2]$
refine te w=64	0.48	0.35	0.63	not generated
Gemma-4B en w=16, blind	0.68	0.31	1.00	$+2$ $[-4.1, +6.6]$
Gemma-4B bn w=16, blind	0.59	0.30	0.99	-13 $[-14.9, -5.4]$
Gemma-4B te w=16, blind	0.53	0.29	0.97	-3 $[-11.3, +6.3]$
Gemma-4B en / bn / te w=64	0.80 / 0.72 / 0.60	0.30	1.00	—
Gemma-12B te w=16, blind	0.57	0.30	0.95	-7 $[-8.9, -0.3]$
Gemma-12B te w=64	0.62	0.30	1.00	-5 $[-5.0, +0.2]$

At the answer span. Which tokens does a $+0.07$ gain in passage retention buy for $+19$ points? A preregistered arm (`iclr-answer-span-preregister.md`) re-ran the retention hook on the same items with a fourth span, the first gold answer string inside the gold passage (every item carries one; 18 Telugu, 16 Bengali and 3 English tokens on average), at w=64 and $\hat{w}$: 600 prefills (`retain_answer`, Table 21). Two of the three substantive readings missed. The blind ranking keeps the answer span *above* its passage, not below (0.49 vs 0.41 Telugu, 0.53 vs 0.43 Bengali; paired $+0.081$ $[+0.052, +0.112]$ and $+0.100$ $[+0.070, +0.131]$), and the sized window lifts the answer by the same amount as the passage ($+0.074$ against $+0.074$ on Telugu, $+0.097$ against $+0.110$ on Bengali; differences of gains $[-0.017, +0.016]$ and $[-0.029, +0.003]$): the window's gain is spread over the whole gold passage, not concentrated at the answer, and no layer concentrates it either. English keeps its three-token answer at 0.73–0.74, under the registered 0.8, and scores 93–95. The third reading held: pooling the three cells at w=64 with language fixed effects, a logit of correctness on the answer share gives z=4.04 against 2.74 for the passage share (log-likelihood -150.8 against -155.7); fitted jointly the answer share keeps z=3.18 and the passage share falls to -0.87, and the items answered correctly kept more of their answer in every language (0.57 vs 0.44, 0.61 vs 0.43, 0.73 vs 0.67). Exploratory, not registered: the items the window turns from wrong to right are those whose answer span it lifted most (Telugu $+0.124$ on the 21 items fixed against $+0.062$ on the rest; Bengali $+0.153$ on 16 against $+0.087$), and the count of items keeping under a quarter of their answer halves ($20 \rightarrow 11$, $20 \rightarrow 5$); yet with the answer share in the model an

indicator for the sized window keeps most of its coefficient ($0.66 \rightarrow 0.54$, z=2.66), so the kept share of the answer accounts for about a fifth of the window's effect. The retained count of answer keys predicts correctness but does not carry the gain; the preregister's failure branch applies and the token-level mechanism is stated as open.

Table 21: Share of the answer span's keys kept, r=0.75, n=100 per cell (mean / median / tenth percentile over items), beside the gold-passage share of Table 19 and the cell's accuracy.

cell	V	answer tok.	answer kept	gold passage	acc.
te w=64, blind	0	18	0.49 / 0.52 / 0.16	0.41	37
te $\hat{w}$=247	1	18	0.56 / 0.57 / 0.25	0.48	56
bn w=64, blind	0	16	0.53 / 0.56 / 0.19	0.43	57
bn $\hat{w}$=183	1	16	0.63 / 0.62 / 0.30	0.54	71
en w=64	1	3	0.73 / 0.68 / 0.45	0.70	93
en $\hat{w}$=43	1	3	0.74 / 0.69 / 0.46	0.71	95

At matched passage length. The retention table exposes a confound: a Telugu item holds about 14 passages of 685 tokens where an English one holds 39 of 147, so the depth of the blind loss across languages could be passage geometry rather than window geometry. A preregistered arm (`iclr-chunk-preregister.md`) rebuilt the Telugu and Bengali items with every passage, gold and distractor, split at sentence ends into chunks of at most 256 run tokens, the gold chunk holding the answer whole (boundaries may not cross the answer span); 41–42 chunks per 8k prefill, gold chunks of 185 (Telugu) and 201 (Bengali) tokens, no item dropped, 87% of boundaries at sentence punctuation, the same first-100 questions and gold rotation, own baselines (Table 22). Chunking touches neither c nor Q_{90}.

Table 22: Fixed-length passages (256-token chunks), r=0.75, n=100, own baselines; Δpp with fixed/broken counts and exact intervals. The retention columns are the chunked prompts' shares at w=64 (`retain_chunk`); the English blind pads keep 0.54–0.63 of a 147-token gold passage.

	base (orig.)	w=64	$\hat{w}$	q_keep	g_keep	d_keep
te	34 (56)	-11, 7/18, $[-19.0, -0.3]$	-6, 4/10, $[-11.7, +2.3]$	0.96	0.51	0.23
bn	73 (73)	-17^{*}, 0/17, $[-17.0, -10.4]$	-9^{*}, 2/11, $[-12.5, -1.2]$	0.98	0.54	0.24

The registered branches were: A, the hole at w=64 stays interval-cleared below -10 (passage length is a modifier of depth, the framing stands); B, it shrinks into the English blind range (the exposure is restated as two-fold); a middle branch for an interval straddling -10. Bengali lands in A: its baseline does not move (73 to 73), its gold retention at the blind window rises from 0.43 to 0.54 — the floor of the English band — and the hole is -17 ($[-17.0, -10.4]$), where English blind windows lose 2 to 13. Telugu lands in the middle: the point is -11 and the interval $[-19.0, -0.3]$ excludes neither -10 nor near zero, because the chunked baseline is 34 against 56 on the same questions. That drop is not a chunking defect: the gold chunk is intact and answered with a marker in 25 of the 28 items lost, and the loss is positional — front and middle gold falls from 17/34 and 17/33 correct to 8/34 and 8/33 while back gold falls from 22/33 to 18/33 — the uncompressed model losing early gold among forty short Telugu chunks, a retrieval effect this arm did not register and does not explain; Bengali shows no such drop. One testable account: the uncompressed model's Telugu retrieval degrades with the number of candidate segments rather than their length; 512-token chunks (about twenty per prefill) would separate the two, and we leave it open. At $\hat{w}$ the residual is -6 on Telugu (the original -5.7) and -9 on Bengali, worse than its original -1.8; both are reported as measured. What the arm settles: passage length modifies the depth of the blind loss and is not its source — at English-sized passages and English-band gold retention a blind Bengali window still loses seventeen points — and what it leaves open is why a blind Bengali or Telugu window loses more than a blind English one at nearly the same gold-minus-distractor contrast (chunked Bengali $+0.30$ against English's $+0.38$, from $+0.11$ unchunked).

K SCALE: QWEN3-8B

The scale slice covers English and Bengali with the same formula and the same 100 items per cell. English is at ceiling and carries no information: 96 baseline, 97 at the default window, 96 at $\hat{w}$=43. Bengali reproduces the phenomenon: 81 baseline, 64 at the default window (-17 pp, 1 fixed / 18 broken, $p<.001$), and 73 at $\hat{w}$=183, which recovers $+9$ pp against the default window (11/2, $p=.022$) but stays 8 pp below baseline (0/8, $p=.008$). The registered criterion was ± 3 pp, so this is a miss, and we report it as one.

What the residual is not is worth stating precisely, because it is the same diagnosis that applies to Gemma. Of the eight items that the baseline answers and $\hat{w}$ does not, all eight have questions shorter than Q_{90} (38 to 67 tokens), so the window covers the question entirely for every one of them — against 88% fully covered in the pool (Fisher $p=.59$): the residual is, if anything, richer in visible questions than the items it is drawn from, not poorer, which is the direction a visibility failure cannot produce. Seven of the eight reproduce the question verbatim in the generated text; and all twelve items whose question exceeds Q_{90} — the only items where the language-level percentile could still leave the scorer blind — are answered correctly under $\hat{w}$. Among the items that the default window breaks and $\hat{w}$ does not recover, three of seven run to the decode cap without ever emitting an answer marker, against none of the eleven items it does recover, and six of the seven sit at the middle gold position. The errors that remain are near-copies of the gold span (a dropped morpheme, an interchanged grapheme) or answers whose supporting sentence did not survive eviction — ordinary compression damage, not blindness.

L A THIRD FAMILY: LLAMA-3.1-8B-INSTRUCT

Both inputs to the rule are re-measured on the Llama tokenizer, which is costlier than Qwen's on both: the trailing blocks are $25/125/190$ tokens for English, Bengali and Telugu (Qwen: $25/107/167$) and the held-out question percentiles are Q_{90}=$18/87/94$ (Qwen: $18/76/80$), giving $\hat{w} = 43/212/284$. Baselines are $97/76/51$ (Table 23).

Bengali reproduces the hole at the default window — 76 $\rightarrow$ 58, -18 pp (2 fixed / 20 broken, $p=.0001$), with a 95% interval of $[-21.5, -9.2]$; the full-pool Telugu hole is $[-22.4, -17.3]$. Here $\hat{w}$=212 recovers $+10$ pp (13/3, $p=.021$) while remaining -8 pp below baseline ($p=.057$). English is at ceiling ($97/98/98$). Telugu shows no clear hole at the default (-5, ns) and $\hat{w}$ meets the gate (-1), but the cell is low-confidence: 88 of 100 Telugu outputs never emit the answer marker, so scoring rests entirely on the first-sentence rule and the marker-only robustness check that backs the Qwen tables does not exist here. We quote Telugu-Llama only as "no clear hole; the formula does not hurt".

The residual audit is more mixed than at 8B. Of the eleven items that the baseline answers and $\hat{w}$ does not, ten have questions within Q_{90} (the window covers them fully — 91% against 88% in the pool, Fisher p=1.0; seven of the eleven sit at the middle gold position, and 4 of 10 unrecovered default-window breaks ramble to the decode cap), while one belongs to the long decile — the first arm in which the registered percentile leftover is actually visible: 10 of 12 long-question items are correct under $\hat{w}$, against 12 of 12 at 8B.

Table 23: Llama-3.1-8B-Instruct, n=100 per cell, eviction ratio 0.75, own stack. Accuracy, with the paired difference against this model's uncompressed baseline and its exact 95% interval.

	c	$\hat{w}$	baseline	w=64	at $\hat{w}$
en	25	43	97	98 (+1)	**98** (+1; $[-0.9, +1.0]$)
bn	125	212	76	58 (-18^*)	**68** (-8; $[-12.7, +0.2]$)
te	190	284	51	46 (-5)	**50** (-1; $[-5.6, +4.4]$)

M TWO MORE SLICES: QWEN3-32B, AND A 16K PREFILL

Both arms were preregistered and run on fresh pods; each store is self-contained with its own baselines.

Qwen3-32B (own stacks). The shipped integers reproduce on the 32B checkpoint (shared tokenizer and template; the driver re-derived c and Q_{90} on pod both times). Bengali (n=100): baseline 80, the default window leaves an interval-cleared -12.0 pp hole (1/13, $[-13.9, -4.5]$), $\hat{w}$=183 misses the point gate at -5.0 ($[-8.5, +1.8]$) with a $+7.0$ recovery short of significance (p=.065) — the Bengali residual at every scale above 4B, a third time. Telugu was inconclusive at n=100 (-4.0, 8/12, $[-12.4, +5.6]$), so a final preregistered arm applied the everything-there-is rule: on the full 669-item pool the hole is interval-cleared at -10.2 (35/103, $[-13.1, -6.8]$, $p=6\times10^{-9}$) — the n=100 cell was sampling noise, the depth arm's own 4B lesson repeated — and $\hat{w}$ meets the gate at -1.6 (16/27, $[-3.5, +0.4]$) with an interval-cleared $+8.5$ recovery ($[+5.4, +11.1]$). Where the 4B pool leaves an interval-cleared -5.7 residual, the 32B pool leaves -1.6 with zero inside the interval: a smaller residual, whose interval still admits -3. The broken items concentrate mid-document (46.6% vs 33.3%, p=.003), replicating the oracle arm's position fingerprint on an independent model and pod. One scoring caveat, registered before the data: 32B answers in prose and copies the instruction's placeholder verbatim after the marker, so marker compliance is 0–1% and the marker-only check does not exist for either 32B arm; every number rides the registered lenient scorer's prose branch.

16k prefill (Telugu, n=100, the same first-100 qids). The blind mode is prefill-independent, as the threshold account requires: at 16k the default window loses -15.0 (3/18, $[-19.7, -5.7]$, p=.0015) from a baseline of 54. The *same* $\hat{w}$=247 — no re-measurement, both inputs are prefill-free — meets the gate (-2.0, $[-6.6, +4.1]$) and recovers $+13.0$ (15/2, $[+4.6, +16.5]$, p=.0023). The 8k-vs-16k magnitudes (-19 to -20 vs -15) are stated descriptively; the item sets differ in distractor packing, so no cross-length test is run.

N ONE ENGLISH INSTRUCTION FOR EVERY LANGUAGE

Every language-dependent number in the main text comes from translated instructions. Many deployments instead ship one English instruction to all languages, which sets $c \approx 25$ for every language and removes the blind mode by construction — but not partial visibility: at w=64 with c=25 the slack is 39 tokens, below the held-out question medians of Bengali (46) and Telugu (54), so most items still show the scorer only part of their question. A preregistered arm ran this cell: English instruction, Bengali and Telugu questions and passages, instruction-last, its own baselines, n=100 per language, with c re-measured for the cross prompt (25) and the rule's prescription $\hat{w} = 25 + Q_{90}$ alongside.

Table 24: English instruction, non-English questions. Accuracy, with paired Δpp against the arm's own uncompressed baselines and exact 95% intervals; n=100; Qwen3-4B, ratio 0.75. No difference is significant.

	baseline	w=64	$\hat{w}$
bn ($\hat{w}$=101)	79	76 (-3; $[-7.7, +3.6]$)	77 (-2; $[-6.6, +4.1]$)
te ($\hat{w}$=105)	62	58 (-4; $[-7.5, +2.4]$)	58 (-4; $[-4.0, +0.8]$)

Neither language shows a significant loss at the default window, and the registered enrichment test has nothing to find: 70 (bn) and 93 (te) of 100 items have $V < 1$ at w=64, the cells break 6 items each, and the V<1 share among broken items matches the pool (Fisher p=.67 and 1.0). This is the registered scope-narrowing branch: partial visibility with an English-sized tail is benign on these items — consistent with English at w=32 and with Gemma's undamaged blind cells — and the damage the paper documents requires a trailing block long enough to hide the question *entirely*. The n=100 intervals cannot rule out losses near 7 pp, and the prompt family here is new (English instruction, non-English passages), so these six cells are never pooled with the frozen-instruction stores. One robustness note: under the stricter marker-only scorer the Telugu cells lose 12–15 pp — compression damages the #### formatting convention in this cross-lingual prompt while the answer content largely survives, a formatting fragility we report here and do not carry into any headline number.

O RIVAL REMEDIES, AND A SECOND MEMBER OF THE FAMILY

Two remedies compete with sizing the window: switch to a scorer that never reads the question, so that nothing can be hidden from it, or ship one larger constant and stop measuring. Both were run on the same items, at matched ratios, against each store's own uncompressed baseline (Table 26). A third block asks the complementary question: whether the integer we compute is a property of the prompt or of the method it was first tested on.

Scorers that do not read the question. Expected Attention ranks by an estimate of future attention (Devoto et al., 2025) and TOVA by attention already paid (Oren et al., 2024); neither can be blinded by a trailing block, because neither was looking. At $r{=}0.75$ both lose significantly in every language — including English, where the trailing block is 25 tokens and no window we tested costs anything. Their damage does not follow c, which is what distinguishes it from the failure this paper describes; it is simply the price of ranking a cache without knowing what is being asked. Blindness is a property of a window that can be sized; query-agnosticism is a property of the method.

One prediction registered for this arm cannot be scored as written: it compares Thai against Bengali, and the arm as locked runs English, Bengali and Telugu. We flag the drafting error rather than substitute a different comparison after the fact. The claim it was meant to test — that this damage does not track c — is carried by the English column, which was written into the same preregistration.

One larger constant. At $r{=}0.9375$ the retained cache is about 512 tokens on these 8k prompts, so a $w{=}256$ window protects half of what survives. It costs English 9 pp against baseline ($p{=}.049$; against the default window the same gap is -7 and does not reach significance at $n{=}100$), while $\hat{w}_{\text{en}}{=}43$ costs one. It does not beat $\hat{w}$ anywhere: -8 pp on English ($p{=}.077$) and -4 pp on Bengali (ns), so the dominance is carried by English and we do not claim it per language. Bengali collapses at the default window here (-44 pp) and $\hat{w}$ recovers $+20$ pp of that (22 fixed / 2 broken, $p{<}10^{-4}$), its largest recovery at this ratio — and still finishes 24 pp below baseline. Sizing the window is what removes the blind mode; it is not what makes aggressive eviction free.

The open question is the other ratio: does the constant suffice where the headline numbers live? Measured on its own pod and baselines, judged by the same preregistered ± 3 gate the rule was judged by, $w{=}256$ at $r{=}0.75$ is $+1$ on English, -3 on Telugu — at the boundary — and -5 on Bengali (5 fixed / 10 broken, ns, CI $[-11.5, +3.5]$), missing the gate that $\hat{w}$ met at -2. The preregistration expected the constant to pass and committed to rewriting our claim if it did; it did not, and we state the symmetric caveat instead: at $n{=}100$ the Bengali interval settles neither the miss nor a close, so what the grid supports is this: at the headline ratio, on its own pod and baselines, the constant meets the gate on English, sits at the boundary on Telugu, and misses it on Bengali — the gate the rule met on the campaign pod; at $r{=}0.9375$, where the two run on one pod, the constant trails the rule on both languages (-9 vs -1 on English, -28 vs -24 on Bengali) and beats it nowhere they share a pod.

The constant at scale. The grid above is bounded by its pod and its hundred items, so we preregistered a re-run at the full validation pools (`iclr-constant-depth-preregister.md`; registered and externally timestamped before any row existed): $w{=}256$ beside the shipped $\hat{w}$ integers, on the campaign stack, with baselines byte-identical to the depth arm's (782/782) — so unlike the bracketed cells above, every comparison here pairs within one stack. The registration fixed the Telugu reading in advance: the constant sits nine tokens above the computed integer ($256 = \hat{w}_{\text{te}}{+}9$), so no separation was expected there, and none was found — the head-to-head is $+1.2$ pp in $\hat{w}$'s favour (16 fixed / 8 broken, CI $[-0.4, +2.5]$): a tie, with zero inside the interval by four tenths of a point. Where the two windows genuinely differ, Bengali, the point gate splits: $\hat{w}$ meets it (-1.8) and the constant misses it (-3.5, CI $[-9.9, +4.1]$), with a head-to-head whose interval includes zero ($+1.8$, CI $[-3.6, +5.9]$) on a pool its size caps at 113. Against its own baseline the constant's Telugu residual is interval-cleared at -6.9 ($[-9.0, -4.2]$), deeper than $\hat{w}$'s -5.7: a 256-token protected tail costs slightly more than a 247-token one and recovers nothing. We claim no interval-cleared separation at this ratio. What the full pools support: the constant misses both of its point gates where the rule misses one, its point estimate trails the rule's in both languages, and it comes closest exactly where it reproduces the rule's own integer plus nine.

The window across ratios. A preregistered grid (`iclr-ratio-grid-preregister.md`; store `ratio_grid`, own baselines, byte-identical to the campaign's on all 200 shared rows) ran Telugu at r=0.5, 0.875 and 0.9375 and Bengali at 0.5 and 0.875, each at the default window, $\hat{w}$ and w=256, n=100 (Table 25). Its registered predictions split, and we state both halves. Held: the default's loss is monotone in the ratio (Telugu -4, -19, -33, -42 and Bengali -4, -16, -30, -44 across 0.5–0.9375), the residual at $\hat{w}$ shrinks as the ratio falls (Telugu $+1$ at 0.5, gate met), and $\hat{w}$ is nowhere an interval-cleared loss at 0.5. Missed: the default's loss is *not* significant at r=0.5 in either language (-4; p=.29 and .39) — a blind ranking that keeps half of everything keeps the gold within noise at n=100; Telugu's recovery at 0.9375 is $+8$ (13/5, p=.096), under the registered $+10$; and the constant is never separated from the rule on either language at any ratio (head-to-heads $+2$, -2, $+1$ on Telugu and $+1$, $+5$ on Bengali, none significant), so the separation reported above is English-only. The arithmetic at 0.9375 is the window tax in its Telugu form: about 512 tokens survive, a 247-token protected window is half of them, and what protecting it can buy is capped by the $\sim$265 left for passages; Bengali, with $\hat{w}$=183, recovers $+20$ at the same ratio.

Table 25: The window across eviction ratios (`ratio_grid`): paired Δpp against the store's own baseline (Telugu 56, Bengali 73), n=100, exact 95% intervals; star: McNemar $p < .05$; head-to-heads paired within the store, with fixed/broken counts. Bracketed rows are the campaign's own cells at 0.75 (Tables 2 and 3; @669 and @113 are the full pools, Appendix E) and Bengali at 0.9375 (`ratio`, above), quoted as context.

	r	default w=64	$\hat{w}$	w=256	$\hat{w}$ vs default	$\hat{w}$ vs 256
te	0.5	-4 [-7.5, $+2.4$]	$+1$ [-3.5, $+4.5$]	-1 [-4.5, $+3.5$]	$+5$ (7/2)	$+2$ (3/1)
	[0.75]	[-19^*]	[0; -5.7@669]	[-6.9@669]	[$+19^*$; $+14.5$@669]	[$+1.2$@669]
	0.875	-33^* [-36.5, -23.5]	-15^* [-19.7, -5.7]	-13^* [-18.7, -3.4]	$+18^*$ (20/2)	-2 (0/2)
	0.9375	-42^* [-46.7, -31.5]	-34^* [-38.7, -23.7]	-35^* [-39.7, -24.7]	$+8$ (13/5)	$+1$ (5/4)
bn	0.5	-4 [-9.6, $+3.6$]	$+2$ [-3.3, $+5.5$]	$+1$ [-3.5, $+4.5$]	$+6$ (8/2)	$+1$ (3/2)
	[0.75]	[-16^*]	[-2; -1.8@113]	[-3.5@113]	[$+14^*$]	[$+1.8$@113]
	0.875	-30^* [-34.7, -19.8]	-12^* [-20.0, -1.2]	-17^* [-21.7, -7.5]	$+18^*$ (21/3)	$+5$ (7/2)
	[0.9375]	[-44^*]	[-24^*]	[-28^*]	[$+20^*$]	[+4]

A tail past the constant. The constant ties the rule above because every measured c is below 256. A preregistered cell built the case it cannot cover with the paper's own pad construction on the primary model: the Telugu instruction padded with a content-free Telugu filler to c=270 (measured on the run tokenizer), $\hat{w}$=350, n=100, own baseline 57 (store `te_tail`). At w=256 every item is blind and the cell loses -20 pp (2 fixed / 22 broken, [-23.5, -11.0]); at $\hat{w}$ it loses -6 (1/7, [-7.9, $+0.4$]), missing the registered ± 3 point gate by the size of the residual the full pool establishes elsewhere; the head-to-head favours $\hat{w}$ by $+14$ (16/2, [$+5.5$, $+17.5$], p=.0013) — the one cell in this paper where the rule and the constant separate on a damaged language, by construction. Marker-only scoring keeps the ordering (-21 / -12).

The question repeated at the end. The practitioner's first fix — append the question again after the instruction, so that the window holds a copy — was run as its own layout (`passages + question + instruction + question + suffix`; store `qdup`), Telugu and Bengali, n=100, against its own uncompressed baseline. At the default window it loses -7 on Telugu (1/8, [-8.9, -0.3], p=.039) and -6 on Bengali (4/10, [-11.7, $+2.3$], ns): the blind mode removed where the prompt can be appended to, an eviction residual left, the same profile as the question-last layout (-6 / -4 at n=100). The registered expectation that items with questions longer than the 59 tokens the window can hold would fare worse did not show: those items are less damaged in both languages (-3.8 against -8.1 on Telugu, n=26 vs 74; -2.9 against -7.6 on Bengali). One unregistered observation: the duplicated prompt's *uncompressed* accuracy is 63 on Telugu against 56 for the original layout on the same items, and 75 against 73 on Bengali — repeating the question helps the uncompressed model, so that at the default window the duplicated prompt lands exactly on the original layout's uncompressed score.

No ranking at all. The same store carries the control the registry always reserved: `random@r0.75`, eviction with no scorer. It does not lose answers; it collapses generation — 6/1/0% accuracy on English, Bengali and Telugu, with 96–100% of outputs running to the decode cap, mostly as repetition loops. Even a blinded ranking is carrying enormous weight. Because

random sits at the floor for every language, it bounds the value of ranking from above without discriminating between them, and we draw no per-language conclusion from it.

A second member of the family. PyramidKV (Cai et al., 2024) keeps the observation window and changes what is done with the retained budget, allocating more of it to lower layers; its kvpress implementation subclasses SnapKV and inherits w=64 unchanged. On its own pod and its own baselines, that inherited default opens holes of -28 and -32 pp — PyramidKV's own numbers, deeper than the -16 and -19 the campaign pod recorded for SnapKV; SnapKV was not run on that pod, so the contrast is between methods and is quoted as context, not read as a pod effect — and the AutoWindow integers, used exactly as shipped, remove them: -2, -1 and $+0$ on Bengali, Telugu and English, with recoveries of $+26$ (29 fixed / 3 broken) and $+31$ (32/1), the two largest in this paper. We registered the opposite guess about the depth, expecting the pyramid allocation to shallow the blind hole rather than deepen it, and report the outcome as an observation we did not predict. The transfer itself is the point: an integer computed from a tokenizer and a template, never tuned against accuracy on either method, repairs both.

Table 26: Rival remedies, paired Δpp against each store's own uncompressed baseline, n=100 per cell; star: McNemar $p < .05$. First block: two query-agnostic scorers at the ratio used throughout the paper, beside the windowed cells they would replace. Second: the heavy-ratio sweep, where $\hat{w}$ is 43 for English and 183 for Bengali. Third: the constant and the no-ranking control at the headline ratio. Fourth: PyramidKV. Both ran on pods that did not reproduce the campaign stack, so their bracketed SnapKV values are the campaign's, quoted as context rather than comparators (SnapKV was not run on those pods) and never pooled with them. Bold: the rule's window.

r=0.75	Exp. Attention	TOVA	SnapKV w=64	SnapKV $\hat{w}$
en	-14^*	-8^*	$+0$	$\mathbf{+2}$
bn	-18^*	-16^*	-16^*	$\mathbf{-2}$
te	-20^*	-18^*	-19^*	$\mathbf{+0}$
r=0.9375	w=64	w=256	$\hat{w}$	
en	-2	-9^*	$\mathbf{-1}$	
bn	-44^*	-28^*	$\mathbf{-24}^*$	
r=0.75, own pod	random	w=256	[SnapKV $\hat{w}$]	
en	-87^*	$\mathbf{+1}$	[+2]	
bn	-72^*	-5	[−2]	
te	-56^*	$\mathbf{-3}$	[+0]	
PyramidKV	w=64	$\hat{w}$	[SnapKV: w=64 / $\hat{w}$]	
en	$+2$	$\mathbf{+0}$	[+0 / +2]	
bn	-28^*	$\mathbf{-2}$	[-16^* / −2]	
te	-32^*	$\mathbf{-1}$	[-19^* / +0]	

The library's own pipeline. kvpress's text-generation pipeline compresses a context string and appends the question uncompressed afterwards, so its window never holds the question in any language. A preregistered arm (`iclr-ctxq-preregister.md`; store `ctxq`, its own pod, one stack) ran that path on the frozen instruction-last items of all eight languages at r=0.75: the same prompt byte for byte, compression stopping at the question's first token, the tail ($c + |Q_i|$, 42–226 tokens) prefilled uncompressed at its original positions, greedy decoding to EOS — the pipeline's own procedure, reimplemented so that the store records every count. An uncompressed twin of the two-phase path is the baseline; it reproduces the single-pass baseline within ±2 pp in every language (72–90% of outputs byte-identical; the rest is bf16 kernel noise between one prefill and two). Table 27: the split path loses in every language, English included, every interval far below zero, and deeper than the two query-agnostic scorers above (English -29 against -14 and -8): a window that ends inside the last passage ranks the cache by that passage's tail, a distractor two times in three. What this paper measures is what single-string prompting *adds* to that floor: a window that reaches the question restores English and leaves Bengali and Telugu blind. The registered pre-condition that the two paths be byte-identical on two smoke items fired (1 of 2) and was restated before the store's

first row (amendment A1: same length, clean split, same score); the byte-identity share is reported in the table as measured.

Table 27: The context-then-question path (`ctxq`; Qwen3-4B, r=0.75, n=100, one pod). Accuracy under the lenient rule; twin = the uncompressed two-phase path; ident = share of twin outputs byte-identical to the single-pass baseline; paired Δpp of the split press cell against the twin, fixed/broken, exact 95% interval; kept = mean cache entries after compression; tail = mean tokens after the split ($c + |Q_i|$).

	baseline	twin	ident	press	Δ vs twin	95% CI	kept / tail
en	93	92	90%	63	−29 (3/32)	[−33.7, −18.9]	2019 / 42
zh	83	84	89%	61	−23 (4/27)	[−28.7, −12.5]	2013 / 47
es	88	88	88%	61	−27 (1/28)	[−28.9, −18.7]	2011 / 59
vi	83	81	72%	55	−26 (1/27)	[−27.9, −17.7]	2017 / 60
th	85	85	88%	43	−42 (1/43)	[−43.9, −33.4]	2016 / 81
sw	56	56	76%	32	−24 (1/25)	[−25.9, −15.8]	2001 / 67
bn	73	74	86%	41	−33 (2/35)	[−36.5, −23.5]	2001 / 164
te	56	58	77%	27	−31 (2/33)	[−34.5, −21.6]	1985 / 226

P A SECOND CONSTANT: THE DECODE CAP

The main arms decode under a 384-token cap, and Appendix R says why in a sentence; this appendix is the measurement behind it. The campaign's first arms used 128, and a decode cap is itself a token-denominated constant — so the paper's thesis applies to it, one layer further out, in the evaluation harness rather than in the evictor. The evidence comes from the cap-128-era stores: a different stack and the byte-parallel item variant, whose inputs are byte-equal across languages. Nothing here is pooled with the main tables, and every number is recomputed by `decode_cap_ledger.py`.

At a 128-token cap, the share of *uncompressed* baseline generations that hit the cap runs from 1% in Chinese to 65% in Greek (Table 28) — a spread produced by one integer, on byte-equal inputs, with no compression anywhere in the picture. Because the instruction asks for the answer marker at the end of the reply, truncation does not merely shorten an answer: it decides which branch of the scoring rule fires. At 128, 95% of English baselines carry the marker against 45% of Greek; at 384 the marker comes back (90/97/92% for Greek, Hindi and Russian). An English-validated harness would see none of this.

Greedy decoding makes the 128-token output a byte prefix of the 384-token output on 1,124 of the 1,200 generations the two runs share, so the cap's effect is isolable within an item rather than inferred across conditions. On the 327 generations where truncation destroys the marker, our scoring rule loses 4 pp (19 fixed / 7 broken, p=.029; Greek +2 and Hindi +1, both ns, Russian +11, p=.039). That is real and it is bounded, because the rule's prose-fallback branch was built to absorb exactly this failure. We cannot say from these stores what a format-strict harness would lose — our own strict marker path also fails for reformatting reasons that have nothing to do with the cap — so we claim only the part we can measure: the cap selects the scoring branch per language, and under our rule the residual cost is four points on the cells it touches. This is why the main arms decode at 384, and why we treat every token-denominated constant in the pipeline — window, cap, budget — as a quantity to measure rather than a default to inherit.

Table 28: The decode cap as a second token-denominated constant. Uncompressed baselines on byte-parallel items, n=100 per language, from the cap-128-era stores; Greek, Hindi and Russian were re-run at 384. Truncated: share of generations that reach the cap. Marker: share carrying the end-of-reply answer marker that the scoring rule keys on.

	el	hi	ru	th	de	en	vi	es	zh
truncated at 128	65%	50%	30%	14%	12%	6%	4%	4%	1%
marker at 128	45%	59%	71%	58%	74%	95%	94%	96%	98%
truncated at 384	17%	4%	2%	—	—	—	—	—	—
marker at 384	90%	97%	92%	—	—	—	—	—	—

Q FROZEN INSTRUCTIONS AND PROMPT LAYOUT

The QA instructions are one line per language, frozen under a prompt version key so that a change to them would invalidate the stored runs rather than silently alter them. They carry the same communicative content — answer from the passages above, end with a marked span — and differ only in how many tokens the model's tokenizer needs for it: 20 for English, 102 for Bengali, 162 for Telugu on Qwen3-4B. Every instruction-last item has the layout

```
[gold + distractor passages, packed to 8k tokens]
<question>
<instruction>
<chat suffix>          # Qwen: <|im_end|>\n<|im_start|>assistant\n
```

and the observation window is the last w tokens of that concatenation. In English, for instance:

```
... Paris is the capital of France. ...
What is the capital of France?
Answer the question using only the passages above.
End your reply with '#### <exact answer span>'.
<|im_end|>
<|im_start|>assistant
```

The padded conditions prepend content-free filler to the instruction so that the marked-span sentence stays adjacent to generation and only the token count changes; the schema conditions prepend a JSON response specification in the same position.

R MEASUREMENT TRANSCRIPTS AND RUN PROVENANCE

Stores. Forty-seven stores back the main runs and their controls: forty-two generation stores — `cliff_multi`, `cliff_en`, `autowin`, `autowin_q90`, `autowin_8b`, `cliff_gemma`, `gemma_q90`, `schema`, `schema_fix`, `v_trace`, `v_trace_bn`, `llama`, `instr_first`, `agnostic`, `ratio`, `pyramidkv`, `constant`, `depth`, `constant_depth`, `slack_depth`, `if_depth`, `thsw`, `xinstr`, `refine`, `oracle_depth`, `ctx16k`, `qwen32b`, `qwen32b_depth`, `w32`, `stuffed`, `pad_base`, `ratio_grid`, `slack_q50`, `te_tail`, `qdup`, `chunk`, `oneruler`, `public_tail`, `gemma12b`, `xstack_qwen`, `ctxq` and `noq` — and five prefill-only stores — `retain`, `retain_chunk`, `retain_oneruler`, `retain_answer` and `retain_layouts`. Appendix P draws on two further cap-128-era stores, `e3` and `e3_384`: forty-nine in all, and every number in the paper is recomputed from them.

Registration timing. For nineteen of the original twenty-two arms the version-history timestamp of the registration precedes the first generation in the corresponding store; the other three — the Q_{90} re-measurement, the 8B slice and the schema re-run — entered version control in a later batch. For the constant-at-scale arm the public anonymised release carrying its registration was pushed 48 minutes before its first generation, so a publicly visible external timestamp precedes every row it governs. For the twenty-four later arms, registered in eleven batches after earlier results existed, the registrations were pushed to the hosting server before each batch's first generation — over nine hours, over an hour, 43, 73, 15, 39, 35, 18, 26, 13 and 20 minutes ahead, by batch, and the one amendment (the context-then-question arm's smoke gate) nine seconds before its resumed block started and before the store's first row; third-party server timestamps that precede every row — but to the private working repository: the public anonymised release that carries them was rebuilt only after their results existed, and we state that distinction rather than blur it.

Stacks. Each generation record stores a stack descriptor (GPU, driver, CUDA, torch, transformers, kvpress) and every comparison in this paper is within one descriptor. The Gemma cells share one stack and the Qwen3-8B slice another; the Qwen3-4B cells share a third, with one exception — the PyramidKV and constant-control pods each landed on descriptors of their own (a newer driver patch or a different host kernel; same GPU model and library versions), so those arms pair only within themselves and are quoted beside the others rather than pooled with them. We never pool across descriptors.

Determinism, measured. The follow-up runs were executed weeks after the original ones, on new machines, and they re-ran cells that the earlier stores already contained — uncompressed baselines, default-window cells, the rungs a later ladder shares with an earlier one. We do not count these by hand: `determinism_ledger.py` keys every generation by model, task, language, treatment, item and prompt version, takes the earliest run of each key as the original, and compares every later run against it. It finds 10,301 generations produced a second time in a later store, and all 10,301 reproduce the original text byte for byte — the largest single block being the pad-baseline store, whose twenty compressed cells repeat all 2,000 rows of the original English pad sweep exactly, and the slack-ablation and Q_{50} stores each repeat the depth arm's 1,338 corresponding rows. This is what allows us to treat the re-measured constants and the earlier tables as describing the same system. Three hundred of those pairs also cross a naming difference — one store requests the default window implicitly and the other names $w{=}64$ explicitly — and agree token for token, as they should. The drifted pods bought a measurement we had not planned: 3,238 generations are shared across stack boundaries — the PyramidKV and constant-control baselines; the oracle arm, which reproduces the depth arm's 1,338 corresponding rows byte-for-byte from a different pod; and the 32B depth arm, which reproduces its own $n{=}100$ slice's 300 rows across a third pod boundary; and the context-then-question arm, whose 800 single-pass baselines reproduce the campaign's on a fifth descriptor — and every one is byte-identical across the boundary it crosses. A deliberate repeat closes the question for the headline cells: the Telugu and Bengali baseline, $w{=}64$ and $\hat{w}$ cells regenerated on an L40S (`xstack_qwen`, a fourth descriptor) return deltas within two points of the campaign's (-21 and -2 against -19 and 0; -17 and -3 against -16 and -2) with 59–85% of outputs byte-identical across the GPU family. The ledger counts same-descriptor repeats (10,301, the figure above) and cross-descriptor ones (3,238) separately, and the argument that the earlier and later tables describe one system rests only on the former. Two earlier pod attempts of the final arms died partway (a guard abort, a container loss); their partial stores, on descriptors of their own, were never candidates for analysis, and the completed arms were regenerated in full on the campaign stack.

Decode cap. Every arm reported in the main text uses a 384-token decode cap. The supporting window-widening comparison in Appendix S was measured under a 128-token cap, as were the historical instruction-first numbers kept beside it; the instruction-first layout swap itself has been re-measured at the current cap and is reported in the main text (Table 3). We flag the difference here rather than leave it to be discovered in a log. The cap matters, and it is an instance of this paper's subject rather than a detail of it: at 128 tokens the share of answers truncated varies from 1 to 65% across languages, which is why the main arms were re-run at 384 (Appendix P).

S TWO FURTHER LEVERS ON VISIBILITY

Two earlier interventions move the same quantity and are consistent with the account in the main text. Both were measured under the 128-token decode cap noted above, so we report them as supporting rather than as headline evidence; the layout swap has since been re-measured at the current cap and moved to the main text. The earlier cap-128 measurement is retained here only as provenance: Bengali $+14$ pp at a fixed ratio (34/6, $p{<}10^{-5}$, $n{=}200$) and $+22$ pp under an absolute cache budget (51/7, $p{<}10^{-8}$, $n{=}200$), with a Thai cost of 6.5 pp. The current-cap rerun locates that Thai cost in the uncompressed baseline (a layout effect, -5 pp, ns) rather than in compression, and notes that Thai drops the answer marker in 43% of question-last outputs, so part of the shift is format behaviour. Widening the window from 64 to 128 tokens at a fixed ratio recovers Bengali by 10.5 pp (22 fixed / 1 broken, $p{<}10^{-5}$, $n{=}200$) and moves no further in these pairings: 256 tokens adds nothing detectable (-0.5 pp, 8/9). The dose ladder, at the current cap and $n{=}100$, still gains from slack 16 to 32, so we read this cap-128 pairing as supporting the coarse shape only.

T THE LAYOUT IN THE WILD

Our packing is ours, which invites a fair question: does any widely used prompt actually put a long block after the question? We surveyed 22 templates selected by popularity and officialness *before* their bodies were read — the default QA prompt of a framework above 5k stars, an official vendor documentation example, or the prompt of a published benchmark or harness — so that safe layouts

could not be filtered out. Each was recorded verbatim from a commit-pinned source, the list was frozen, and only then was anything tokenized. We measure the *static* trailing block: the text after the question placeholder with every unfilled placeholder deleted, on the Qwen3-4B tokenizer. Deleting placeholders makes every number a lower bound, because runtime content only adds (Table 32).

The survey measures geometry; for its strongest single-shot example the damage is now measured too. A preregistered arm rendered the shipped refine template (T06) verbatim over our items — query first, a fixed existing-answer stub, passages as the new context, so $V{=}0$ for every item at any small window. Total blindness in that layout costs English a non-significant -3.0 ($[-4.9, +2.2]$, $n{=}100$) and Bengali -8.0 ($[-12.7, +0.2]$, $p{=}.057$) — blindness is necessary for the failure, not sufficient (the Gemma blind-English cells again), with the Bengali interval leaving room for a real effect. AutoWindow's output for this layout is a refusal — c is the entire post-question suffix, so no practical window clears it; the remedy is to protect or reorder, not to resize. Whether eviction harms an agent loop mid-trajectory remains untested — the question at step k is not the task at step 0 — and the T22-style templates stay static measurements.

Most of what ships is safe, and that is the first half of the result. Single-shot retrieval-QA defaults leave 4–5 tokens after the question, and their chat variants leave 0–4. Modern vendor structured-output and tool-use APIs carry the schema in an API field rather than in prompt text, so at most a token handful (≤ 6) remains after the task — which falsifies the guess we began with, that in-prompt response schemas would be where long trailing blocks live. Harness QA templates leave 3–5 tokens, bar one at 19/15 (T22).

The blocks appear where a prompt is *iterated*. LlamaIndex's shipped refine template puts the query first and, at its empty-placeholder floor, exactly 64 tokens after it — the kvpress default, to the token, before the existing answer or the new context contributes anything. SWE-agent's default instance template puts 221 static tokens after the problem statement, longer than the longest trailing block among our eight languages, before its first observation arrives. ReAct and the OpenHands loop render the user's question once and append every later tool result after it by construction. How far that goes is measured on public trajectories (Table 29; `measure_agent_tails.py`, 300 trajectories per scaffold, the Qwen3-4B tokenizer, chat-template tokens not counted): OpenHands' shipped SWE-bench prompt carries the same 221-token block as T18 after the issue and passes 256 tokens at its second turn (median 340); the older SWE-agent template carries 658 static tokens and is past 256 before its first action. By the fifth turn the median tail is about 2,000 tokens on either scaffold, by the tenth 5,000–10,000. No static public tail passes 256; every iterated one does by its second turn. The full record, with permalinks and verbatim template text, ships with the code.

Table 29: Tokens after the user's issue at the prefill of turn k on public trajectories (medians). Every trajectory is past 256 from turn 2 on both scaffolds, and from turn 1 on SWE-agent. SWE-agent: `nebius/SWE-agent-trajectories`; OpenHands: `SWE-Gym/OpenHands-Sampled-Trajectories`; 300 trajectories each.

turn k	1	2	3	5	10	20
SWE-agent	658	784	1,071	1,938	5,315	10,830
OpenHands	221	340	457	2,638	10,153	17,364

A public tail on the shipped default. The one public template with a long tail was surveyed and not run, because SWE-agent's task cannot be scored from the prompt; a preregistered arm (`iclr-public-tail-preregister.md`) put its static block — 221 tokens, verbatim, the one placeholder filled with the container convention `/testbed` — after the question of the 100 English campaign items, followed by our answer-format line ($c{=}248$, $\hat{w}{=}266$; 400 generations, own baseline, `public_tail`). The arm's registered pre-condition was a working baseline, within 10 points of the campaign's 93; the uncompressed model scores 81 under this tail (a fifth of its replies open as a repository task), two points under the floor, so by its own rule the arm is void and the paper draws no claim from it. Reported as measured (Table 30): the shipped $w{=}64$, blind by 184 tokens, scores 40, -41 against that baseline ($[-45.7, -30.5]$, 3/44), and its replies keep the QA shape and answer from the wrong passage; $\hat{w}$ scores 75 (-6, $[-9.5, +1.1]$; $+35$ over the default head-to-head, $[+24.7, +39.7]$); $w{=}256$, inside which the tail fits, 79 (-2, $[-8.4, +5.4]$). The reader may weigh the interval; the paper does not.

Table 30: The SWE-agent instance template's static block as the trailing block of the English items, r=0.75, n=100, own baseline. Void by its preregistered baseline floor (83); reported as measured.

	V	acc	Δpp	fixed/broken	95% CI
uncompressed	—	81			
w=64, blind by 184	0	40	−41	3/44	$[-45.7, -30.5]$
$\hat{w}$=266	1	75	−6	2/8	$[-9.5, +1.1]$
w=256, tail fits	partial	79	−2	5/7	$[-8.4, +5.4]$

A public suite on its own prompt. OneRuler (Kim et al., 2025b) is RULER translated into 26 languages by native-speaker annotators, instructions included. Its single-needle prompt is `instruction, <text> haystack </text>, <Question> ..."key" ...</Question>, answer format`, so a localised block — the rest of the question, the "list all" and "if none" clauses, the answer format — follows the query key. Measured from the suite's own template and vocabulary on the Qwen3-4B tokenizer, that block is 49 tokens in Chinese and 52 in English and exceeds the shipped 64 in the other 24 languages (Danish 65, Vietnamese 83, Swahili 105, Hindi 264, Tamil 345); the rule returns exactly 64 for English. Swahili and Vietnamese, safe on our template (c=47/39), are blind on this one. A preregistered arm (`iclr-oneruler-preregister.md`) ran the prompt unmodified — the suite's own template, vocabulary and book, one needle, seven-digit values, RULER's substring scoring — on six languages at 8k, n=100, {baseline, w=64, $\hat{w}$}, with the retention subclass of Appendix J on the needle sentence (Table 31). The registered prediction was the mechanism's, not the threshold's: a needle competes with nothing, so a blind ranking should keep it and nothing should be lost even 281 tokens past the window. Nothing was: every compressed cell reproduces its baseline item for item (0 fixed / 0 broken in all eleven cells), on Hindi and Tamil blind by 200 and 281 tokens as on English and Chinese. The retention says why, with the one gap the arm leaves: the blind window keeps 0.58–0.63 of the needle sentence's keys on Tamil and Hindi — above the 0.41–0.43 it keeps of a QA gold passage, below the 0.8 the preregister guessed for the whole sentence — and that is enough, because the payload is four tokens of digits the model copies from wherever they survive, where a QA answer needs the passage around it. The window still buys retention here (0.63 → 0.69, 0.58 → 0.67 at $\hat{w}$) with nothing left to recover. Swahili's uncompressed model answers "none" on 80 of 100 items and the cell is reported as low-power, as registered. So on a standard suite's own prompt the geometry is not ours and blindness is, again, necessary and not sufficient: the loss follows what the answer needs of the retained gold, and this is the case where it needs little.

Table 31: OneRuler single-needle NIAH on its own localised prompt, Qwen3-4B, 8k, r=0.75, n=100 per cell; accuracy under RULER's rule, and the share of the needle sentence's keys the ranking keeps (mean / median over items; `retain_oneruler`). Every compressed cell has 0 fixed / 0 broken against its baseline. c is the block after the key on this tokenizer plus the chat suffix; $\hat{w} = c + Q_{90}$ over the suite's 100 nouns.

	c	$\hat{w}$	base	w=64	$\hat{w}$	kept, w=64	kept, $\hat{w}$
en	52	64	100	100	—	0.75 / 0.73	—
zh	49	60	99	99	99	—	—
vi	83	104	100	100	100	—	—
sw	105	134	20	20	20	0.74 / 0.73	0.76 / 0.73
hi	264	296	100	100	100	0.63 / 0.62	0.69 / 0.68
ta	345	365	100	100	100	0.58 / 0.53	0.67 / 0.61

Table 32: Static trailing block of 22 popular templates, in tokens on the Qwen3-4B tokenizer, with unfilled placeholders deleted so each number is a lower bound. Zero means the question is last. Selection was frozen before any template body was read or tokenized.

	template	tokens
Single-shot retrieval QA		
T01	LangChain `RetrievalQA`, completion	5
T02	LangChain `RetrievalQA`, chat	0
T03	LangChain hub `retrieval-qa-chat`	0
T04	LlamaIndex text-QA default	4
T05	LlamaIndex text-QA, chat	4
T08	Haystack `PromptBuilder` example	5
T09	RAGFlow chat assembly	0
Structured output and tool use		
T10	OpenAI structured outputs	0
T11	OpenAI function calling	0
T12	OpenAI JSON mode example	6
T13	Anthropic tool use	0
T14–T16	Instructor, Outlines, Guidance	0
Evaluation harnesses		
T20	`lm-eval` XQuAD, English	3
T21	`lm-eval` XQuAD, Arabic	5
T22	Belebele, official / harness	19 / 15
Iterated prompts		
T07	LlamaIndex refine, chat	9
T17	LangChain ReAct, static suffix	3
T06	**LlamaIndex refine default**	**64**
T18	**SWE-agent instance template**	**221**
T19	OpenHands mid-loop	221, then grows (Table 29)

U PRODUCTION SCOPE

Re-checked on 2026-08-18, with permalinks in the supplementary record. vLLM exposes KV-cache dtype and scheduler token budgets, not a SnapKV-style evictor; its default attention backends are dense. TensorRT-LLM implements last-w scoring as an opt-in sparse-attention backend whose configuration default is a 32-token window, and skips it entirely below a prompt-length threshold. SGLang has an open proposal for a sparsity framework which states that it must be default-off and names SnapKV and PyramidKV among its explicit non-goals — so the family we study is, for now, deliberately outside the default path of the two most used open serving stacks. Mao et al. (2026) argue that research evictors rarely ship as production defaults, and we cite that against ourselves: the constant we study is a research default and one opt-in production constant, not something most deployments are running today. The reason to fix it now is that the fix is a measurement rather than a redesign.