IKIGAI IS ALL YOU NEED: NON-GRADIENT REFERENT-LEDGER CONSTRUCTION ESCAPES THE BOUNDED-WINDOW CEILING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient-trained language models pay a fixed, growing compute cost to exploit more data: acquiring new knowledge means reprocessing the entire corpus at full training cost. A constructive text generator built with no neural network, no gradient descent, and no transformer does not share this problem. Its acquisition cost grows sub-linearly with corpus size (wall-clock time $\sim N^{0.51}$ across a 122x corpus sweep, process memory flat at about 50 MB for its language model), and, reversing every gradient-scaling law we are aware of, its per-sentence generation speed and reliability actually improve as the underlying corpus grows, by 3.4–4.8x across independently replicated seeds, simply because richer real evidence makes its constraint-satisfaction search easier rather than harder. The mechanism also closes a structural gap that has limited non-gradient generation: any admissibility predicate evaluated over a bounded context window is provably capped at a regular language (Theorem A), and every surface-marker escape mechanism we constructed and tested needed to reintroduce a fixed convention just to stay decodable (Finding B). Unbounded, non-surface generator memory, an exact-identity referent ledger, escapes both, with structural complexity that keeps growing with the tracked-referent pool rather than saturating. At equal training-corpus size, two independent blind human raters judged this mechanism more natural than a converged gradient-trained transformer baseline on the exact matched corpus at matched diversity (2.31 vs 1.94 of 3, $p = 0.011$), and it matches that baseline on two standard diversity and copying measures under the same matched-corpus condition (Distinct-2: 0.766 vs 0.773; copied 7-grams: 0% vs 0.7%) — all with zero backpropagation, zero GPU requirement, and flat RAM regardless of corpus size for its constant-work language model. One axis it does not yet win is perplexity under an external pretrained model, which by construction rewards statistically generic text; we report it rather than omit it. The same clustering mechanism can also directly organize a pretrained transformer's own embedding weights, the literal parameters rather than a distillation, into categories that tie or exceed our from-scratch results. It does not yet match transformer-scale generation unconditionally, and we present it as a first, concrete step in an active research program toward that goal rather than a finished system.

1 INTRODUCTION

Every dominant approach to open-ended text generation shares one property: incorporating new knowledge *into the generator itself* means paying training-scale compute again (retrieval-augmented and kNN models add a datastore but keep a gradient-trained generator). We ask whether this is inherent, by building a generator that pays no such cost by construction: no neural network, no gradient descent, no transformer. Its knowledge is counts; acquisition is one pass.

In one sentence: bounded local evidence has a proven ceiling; flat, off-page state crosses it for exact long-range identity; counting comes within 0.10 nats of the transformer wherever it has evidence and reaches human-preferred fluency at matched diversity; and the gap that remains is clause-level structure, exactly where the theory says flat state stops.

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.

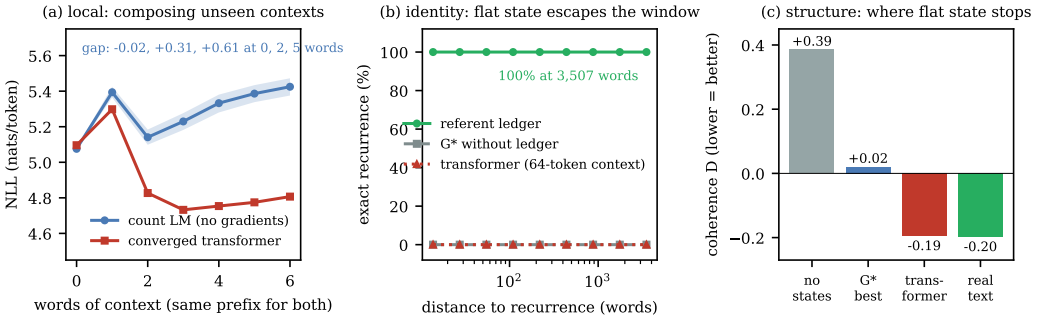


Figure 1: Three capabilities, separated. (a) Local prediction on the clean test set at the first positions of a sentence, where both models see exactly the same prefix (shaded: 95% sentence-bootstrap interval of the gap). Counting is better with no context and loses by composing unseen short contexts; the gap is flat after 5 words. (b) Exact referent recurrence against distance: the flat, off-page ledger at 100%, bounded context at 0%. (c) Cross-sentence coherence D: our best generator narrows the gap from +0.39 to +0.02 but stays above the transformer's and real text's -0.19.

The obvious objection is expressivity: a generator built from attestation checks over n-gram tables should hit the ceiling n-gram models always have. We make the objection precise and prove it (Theorem A and two corollaries), identify the one kind of state that escapes it without a fixed convention (a flat, off-page referent ledger), and show that the kind of state which would be needed for full clause structure (a stack) is exactly the kind that cannot be supplied that way. We then push the counting side as far as we can against a converged same-corpus transformer, on a decontaminated held-out set. The result separates three things that are usually conflated: **local prediction**, where counting comes within 0.10 nats of the transformer wherever it has evidence and loses only by composing contexts it never saw; **persistent identity**, where flat off-page state beats bounded context outright; and **clause-level coherence**, where flat state stops, as the theory predicts (Figure 1).

Our contributions:

- **Theory.** Any bounded-context admissibility predicate is capped at a regular language (Theorem A); none can enforce identity recurrence over an unboundedly growing referent pool (Corollary A.1) or be sound and complete for English syntax at unbounded embedding depth (Corollary A.2). Every surface-marker escape we built needed a fixed convention (Finding B).
- **Local prediction, decomposed.** A closed-form count language model (continuation Kneser–Ney, two-sided low-rank similarity, count bags, EM-fitted latent situations) closes 78% of the held-out likelihood gap to the converged transformer (1.662 $\rightarrow$ 0.369 nats/token). An exact split of the remainder shows where the transformer wins: counting is within 0.10 nats of it wherever the continuation was seen, and loses the rest by composing unseen 1–5-word contexts. Per-token work is constant in corpus size, and process memory stays flat while knowledge grows on disk.
- **A variable-length generator** whose proposals come from attested contexts under a copy budget calibrated to the transformer's own copying, steered by order-free count states. Judged by the transformer, its fluency matches the transformer's own samples at near-equal diversity; judged by two blind humans at matched diversity, it is preferred over them (2.31 vs 1.94 of 3).
- **The referent ledger inside that generator**: exact recurrence at every tested distance to 3,500 words at essentially no fluency cost; bounded baselines 0–2%.
- **An honest limit.** Cross-sentence coherence, measured by how much the transformer's full context helps it predict a text, stays positive for our text (+0.02) against -0.19 for real text; 17 count and closed-form readers of the defect (generative and discriminative, local and whole-prefix, cross-fitted) are at chance, and more data barely moves it. We explain why and tie it to Corollary A.2.

2 Related Work

Expressivity. Self-attention cannot compute periodic or nested dependencies without growing depth (Hahn, 2020); saturated transformers lie in TC^0 (Merrill et al., 2022); transformers degrade on Dyck-style nesting (Bhattamishra et al., 2020). Theorem A is an instance of strictly locally testable languages (McNaughton & Papert, 1971) and Myhill–Nerode (Myhill, 1957; Nerode, 1958; Hopcroft & Ullman, 1979); we claim no novelty for it, only its corollaries for this design space and a mechanism satisfying what they show necessary.

Count language models and non-gradient generation. Kneser–Ney smoothing and its modified form (Kneser & Ney, 1995; Chen & Goodman, 1999), trigger pairs (Rosenfeld, 1996), aggregate Markov models fitted by EM (Saul & Pereira, 1997), spectral learning of latent-state models (Hsu et al., 2012), dependency grammar induction (Klein & Manning, 2004), and randomized SVD (Halko et al., 2011) are the closed-form tools we build from. Corpus-statistics generators rank lattices or overgenerate from grammars (Langkilde & Knight, 1998; Bangalore & Rambow, 2000; Varges & Mellish, 2001); constraint-based text planning (Power, 2000; Mellish et al., 1998; Marciniak & Strube, 2005; Mackworth, 1977) and neural constrained or retrieval-augmented decoding (Hokamp & Liu, 2017; Lu et al., 2021; Lewis et al., 2020; Khandelwal et al., 2020) presuppose a hand-built grammar or a gradient-trained generator. None enforces unbounded-distance identity.

Entity tracking and evaluation. Centering (Grosz et al., 1995) and entity grids (Barzilay & Lapata, 2008) model local coherence; neural checklists and entity vectors (Kiddon et al., 2016; Ji et al., 2017) track entities with gradient-trained continuous state; our ledger is a discrete table. Perplexity rewards generic text (Holtzman et al., 2020), so we compare fluency only at matched diversity (Distinct-2, Li et al., 2016; Self-BLEU, Zhu et al., 2018).

3 Theory: the bounded-window ceiling

Setup. Let Σ be a finite vocabulary. An *admissibility predicate of radius* k is a boolean A_k deciding whether token w_i of $w = w_1 \dots w_n$ is licensed given only $w_{i-k}, \dots, w_{i+k}$; bigram, trigram and skip-gram attestation at offsets up to ± 5 are instances with $k = 5$. Let $L(A_k) = \{w \in \Sigma^* : A_k \text{ holds at every position}\}$.

Theorem 1 (Theorem A)**.** *For any admissibility predicate A_k of finite radius k, $L(A_k)$ is a regular language.*

Proof (sketch; construction in Appendix B). An automaton whose state is the last $2k{+}1$ tokens (boundary-padded) evaluates A_k at each position and rejects on failure; its state count is independent of n. ■ Theorem A bounds the family of finite-radius local predicates, not all non-neural generation: state kept *outside* the predicate is not covered.

Corollary 1 (Corollary A.1)**.** *No admissibility predicate of finite radius can enforce exact reproduction, wherever they recur, of an unboundedly growing sequence of salient referent identities (from a fixed alphabet, repeats allowed) introduced over a passage.*

Proof (sketch; Appendix C). The requirement contains the copy language $\{u\#u\}$ as its intersection with a regular language; the copy language is not regular by the Pumping Lemma, contradicting Theorem A. ■ **Scope.** The ledger enforces the per-referent form: any tracked word must recur at an unbounded distance. No fixed-radius predicate can enforce this, since it never sees both occurrences. The ledger does not recognize the copy language itself; ledger plus window is still finite-state over a fixed vocabulary (Appendix B). What it escapes is the fixed-radius class, not regularity.

Corollary 2 (Corollary A.2)**.** *No admissibility predicate of finite radius can be both sound and complete for English grammaticality at unbounded embedding depth.*

Proof (sketch; Appendix D). Center embedding maps onto $\{a^n b^n\}$ by intersection with a regular language and a homomorphism (Chomsky, 1956). ■ A competence-level result: real text rarely nests deeply, so A.2 says a bounded check can never be *extended* to full recursion, not that shallow sentences are out of reach.

Two kinds of unbounded state. Referent recurrence needs *flat* state (has this word appeared, and which); center embedding needs a *stack*. **Finding B** (empirical): a reserved bracket pair makes

recursive clause splicing exactly decodable (collapse factor 1.0 at depths 1–4 over 800 strings), while every marker-free recursive construction we built showed multiplicatively growing ambiguity (up to $\sim$2000x at depth 4; growth rate $r \in [2.28, 2.78]$ across three mechanisms). A bracket reused identically at every site is a fixed convention, which our design rules out, so flat state is reachable here and stack state is not. Every mechanism below therefore combines bounded-window proposals, order-free (flat) count states, and one flat unbounded memory, the ledger; Section 6 shows the price.

4 MECHANISM

All components are counts over the training corpus and closed-form computations on them (sorting, sums, SVD, least squares, expectation-maximisation). No component is trained by gradient descent; a GPU is used only to batch evaluation arithmetic and to run the transformer baseline.

Count language model. The base is interpolated Kneser–Ney over context orders 1–5 whose lower orders use *continuation* counts (the number of distinct words seen before a gram) and three discounts per order (Kneser & Ney, 1995; Chen & Goodman, 1999). On top of it: (i) σ-2, a two-sided similarity: $P_\sigma(w \mid h) \propto P(w) \exp(\beta\, W_h \cdot C_w)$ from a randomized SVD of the (2-word context $\times$ word) PPMI matrix, mixed in linearly; (ii) a trigger product over every earlier word b of the sentence, $\gamma \sum_b \log \frac{P(w|b)+u(w)}{2u(w)}$ from sentence co-occurrence counts (Rosenfeld, 1996); (iii) a category correction of the spread of probability over words never seen after the deepest matched context (Witten–Bell over 512 count-derived word classes); (iv) latent "situations" fitted by EM: an aggregate Markov model (Saul & Pereira, 1997) and a 512-class model of the 4-word window; (v) a frequency-abstracted backoff table (the 500 most frequent words exact, others their class). The stack mixes all of these with weights set on a dev set; every weight is a single scalar.

Constant per-token work. The base model is rebuilt so that each word costs a constant number of hash lookups (about 10, independent of corpus size) into open-addressing files on disk, with the σ-2 normaliser precomputed per context. Scores are identical to the in-memory model. The knowledge itself grows on disk; what stays constant is the work per word and the process memory.

G*, the generator. Each next word is drawn in proportion to its count after the longest training context of up to 5 words that has an admissible continuation. A word is inadmissible if it would complete a 7-word window attested in training. The budget is calibrated: the transformer's own samples at its human-rated decoding reproduce a training span of median 5 and 90th percentile 6 words (26.5% of its sentences contain a 6-word training span). Order-free count states then restrict the candidate pool, never ranking by argmax: among admissible candidates the most compatible are kept until they hold a fixed share q of the count mass, then one is sampled. The states are an **ending** state (end-of-sentence log-odds from capped counts of 48 count-derived word classes and length, allowed only where the last two words were seen sentence-final), a **topic** bag (co-occurrence vectors of every earlier word, minus the generic direction), an exact **frequent-word** bag (200$\times$200 ordered PMI table), and **far agreement** (the trigger lift of a candidate from words at least 5 back). Optionally the generator draws K whole sentences freely and keeps the one with the highest mean far agreement (best-of-K decoding; no template, slot or frame is involved at any step).

The referent ledger. An off-page set of salient words (outside the 200 most frequent, seen at least 20 times) introduced by the passage, never written into the text. To bring a referent back, a sentence is *anchored* on it: a backward G* (reversed counts, its own ending model and the mirrored states) grows the words before the anchor to a real sentence start, and the forward G* continues to its own ending. No marker, slot or template is involved: the anchor is an ordinary word and both halves follow the same count rules. By Corollary A.1 no bounded-window mechanism can do this; the ledger's state is flat, so Finding B does not apply to it.

5 EXPERIMENTS

Setup. Corpus: 2,115,067 Project Gutenberg sentences. Baseline: a GPT-nano transformer (7.27M parameters) trained on the same sentences for 5.0 h until held-out loss plateaued. **Decontamination:** Gutenberg contains duplicate editions, and 69.7% of held-out sentences contain a 7-word window attested in training. The count model then scores 2.6 nats/token against the transformer's 4.6 by recalling copies. Every likelihood below is measured on the same 4,000 held-

Table 1: Held-out NLL (nats/token) on 4,000 clean test sentences (59,111 tokens, zero training 7-grams; the same set for both models). The last column is the gap to the converged transformer (4.661).

Count model (cumulative)	NLL	gap
Interpolated absolute-discount backoff, orders 1–5	6.334	1.662
+ σ-2 two-sided similarity	5.675	1.014
+ continuation counts in the lower orders	5.300	0.639
+ modified (three-discount) Kneser–Ney, re-tuned σ-2	5.190	0.529
+ trigger bag of earlier words	5.167	0.506
+ category correction, EM situations, abstraction backoff (stack)	**5.030**	**0.369**

out sentences with no attested 7-word window (59,111 tokens) for both models, with weights set on 600 separate clean dev sentences.[1] **Generation metrics** (1,000 sentences per arm), all computed by the converged transformer: NLL per token, Distinct-2, length, and the coherence measure D = mean over word targets $t \geq 7$ of $[s(w_t \mid \text{full prefix}) - s(w_t \mid \langle bos \rangle + \text{last 5 words})]$, where s is surprisal. $D < 0$ means the earlier words help predict the later ones. Every pass/fail bar was written before its run.

5.1 How far counting gets on likelihood

Table 1 shows 78% of the gap closed. The largest single steps are σ-2 and continuation counts.

Identical information. At the first positions of a sentence both models see exactly the same prefix, so no truncation is needed to compare them (Figure 1a; 95% sentence-bootstrap intervals). With no context counting is better (-0.020 [-0.026, -0.014]). With one word the transformer leads by 0.096 [0.066, 0.126], and with two by 0.314 [0.271, 0.357]. By five words it leads by the full 0.612 [0.563, 0.659], and the gap is flat after that. Truncating the transformer to $\langle bos \rangle$ + last k words in mid-sentence puts it at a false sentence start and inflates its loss, so we do not use truncated-transformer scores as evidence of parity.

What remains. We split test tokens by whether the target was seen after the deepest matched context. Where it was (51% of tokens), counting is within 0.10 nats of the transformer. At well-counted 5-word contexts (16.5% of tokens) counting *beats* it by 0.10 nats. 90% of the remaining gap is novel continuations, and 83% of that is *which* unseen word gets the reserved mass: the count model reserves nearly as much probability for unseen words as the transformer does (0.69 vs 0.80). So the gap is local composition, not long-range memory. Fifteen further closed-form devices each moved dev likelihood by at most 0.043 nats. They include longer-context σ (no row exists for an unseen context), per-position pair experts, nearest-context pooling, class contexts, low-rank fold-in of long contexts, a fixed random reservoir read out by ridge or linear-discriminant weights, EM situations, and a spectral automaton (Hsu et al., 2012). One more is *count attention*, the attention operation with query, key and value all taken from counts; it gets monotonically *worse* as it sharpens, because a query built from the model's own prediction amplifies what the model already believes. Data helps with diminishing returns: -0.104, -0.086 and -0.061 nats per doubling from 1/8 of the corpus to all of it.

Runtime. The constant-work model's knowledge is a 5.26 GB store on disk, built in 147 s. A fresh process scoring every clean dev and test token (67,827) needs 51 MB of private memory and grows by 5 MB. It makes 10.3 lookups per token at 102–508 μs per token in plain Python on a CPU (depending on the operating system's file cache; we make no latency claim), and reproduces the in-memory model's likelihood exactly. Most of those 5.26 GB are hash overhead and statistics derivable from the text: the same model also runs from a *compact store*, the training token stream bit-packed plus a suffix array, from which every count, continuation count and context statistic is computed on demand. It takes 255 MB (21$\times$ smaller), builds in about 90 s, matches every probability of the hash store to within 5×10^{-7} relative (dev NLL 5.1442, test 5.162), and runs at 21 μs per token in compiled CPU code; G* on it writes identical sentences (200 of 200 on each of two seeds). Adding

[1] The transformer's plateau validation perplexity (69.5) was measured on uncleaned held-out text; on the clean test set its perplexity is 106 (4.661 nats/token).

Table 2: Generation, 1,000 sentences per arm, all scored by the converged transformer. D: coherence (negative = earlier words help). copy7: sentences containing a 7-word training span. Real text's 69% copy7 reflects duplicate editions.

Arm	NLL	Dist-2	len	>22 w.	D	copy7
Real held-out text	4.511	0.798	14.3	0.0%	−0.197	69%
Transformer, T=1	4.470	0.773	14.4	4.7%	−0.194	0.7%
Transformer, T=0.9, top-40	2.992	0.500	14.1	1.7%	−0.285	6.3%
Sampling the count LM (Table 1, with copy budget)	6.442	0.796	17.4	26.5%	+0.828	0.0%
G*, no states	5.043	0.817	13.8	12.1%	+0.386	0.0%
G* + ending, topic and frequent-word states	4.739	0.794	13.7	0.7%	+0.158	0.0%
+ far-agreement steering	4.654	0.786	13.3	0.3%	+0.124	0.0%
+ best-of-6 by far agreement	4.351	0.766	12.2	0.1%	+0.027	0.0%
+ bilinear-state steering, best-of-6	4.335	0.762	12.1	0.4%	**+0.021**	0.0%
+ future-state steering, best-of-6	**4.310**	0.766	12.2	0.6%	+0.040	0.0%

text grows the files, not the RAM or the work per word. We checked this directly across knowledge sizes. Rebuilt from 1/8, 1/4 and all of the corpus, the store takes 1.05, 1.86 and 5.26 GB on disk. A fresh process scoring every token ends at 48, 48 and 51 MB. Lookups per token are 7.3 and 7.5 for the two subset stores, which omit the bag table, and 10.3 for the full store, where the bag adds a lookup per earlier content word. So memory and work per word do not grow with knowledge; only the files do. The underlying disk hash store keeps its median lookup at 10–19 μs and its process memory at 33–39 MB from 10K to 10M keys. G* itself takes about 40 ms per sentence in plain Python. We have not measured the transformer on the same CPU, so we make no speed claim against it. **Acquisition cost.** On the same corpus and the same laptop GPU, the transformer needed 5.0 h of training (174k steps) to reach its plateau. The count tables behind G* build in about 6 minutes; the continuation tables take 150 s and the constant-work store 147 s. The two EM-fitted situation models of the full stack add about 45 minutes, which is iterative but uses no gradients. Counts are additive, so new text is incorporated by counting it; the low-rank and EM components need a refit, but no generator is retrained.

5.2 Generation

The window caps fluency; copying buys it back. We ran the transformer itself with its context cut to the last k words and scored its text with its full context. It scores 6.33, 5.76, 5.45, 5.26, 5.03 and 4.76 nats/token at $k = 1, 2, 3, 4, 6, 8$, against 4.54 with full context. Every doubling of the window is worth 0.3–0.5 nats, and a 2-word transformer writes no better than a 2-word count sampler (5.40). This is Theorem A measured on generation. Count generators with longer windows reach transformer-level fluency only by copying: an unconstrained 5-word sampler scores 4.52, but 97.7% of its sentences contain a copied 8-word training span. Hence G*'s copy budget.

A better scorer is not a better sampler. Sampling the stacked model draws from its smoothing tail and runs on (Table 2); sharpening it restores local fluency but collapses diversity (Distinct-2 0.48 at NLL 3.74). G* instead samples only attested continuations, so its tail is empty.

Where G* loses fluency, and the states that fix it. Within its own 5-word window, G* text is almost as predictable as copy-matched real text (gap 0.10 nats). The gap (0.52) appears only when the transformer sees the earlier words. It survives shuffling them (0.43), so what is missing is agreement with *which* words came earlier, not their order. Both halves of the vocabulary matter: content words (topic) and frequent words (person, number, tense) each carry about half of it. A counted bag of all earlier words, used alone as a paired chooser, picks the real next word over the window's own alternative 64.1% of the time on content-word decisions. The full transformer scores 57.7% on the same items. The exact frequent-word table scores 60.0% against the transformer's 66.9% on function words; the transformer gains only 4.7 points from context beyond the window. These are the topic and frequent-word states. A plain bag failed its own control, being no better than sharpening at equal diversity; subtracting the generic direction made it the first state off that curve.

Fluency. With the ending state, G* ends its own sentences like real text: mean 13.7 words (real 14.5), 0% at the length cap, 1.1% longer than 22 words, 0.5% ending on a word that almost never

Table 3: What each component buys. Each row adds one component to the row above (likelihood rows: clean test set; generation rows: 1,000 sentences, transformer-judged). Each component moves the capability it targets and little else.

Component	Targets	Measured effect
σ-2 two-sided similarity	which unseen word	LM NLL 6.334 $\rightarrow$ 5.675
continuation counts	lower-order evidence	LM NLL 5.675 $\rightarrow$ 5.300
copy budget (L=6)	copying	8-word copies 97.7% $\rightarrow$ 0% (5-word window)
ending state	sentence length	>22 words 12.1% $\rightarrow$ 0.8%; D +0.386 $\rightarrow$ +0.203
topic and frequent-word bags	agreement with earlier words	+0.16–0.22 nats vs. sharpening at equal Dist-2
far agreement + best-of-6	far context	D +0.158 $\rightarrow$ +0.027
referent ledger	identity at distance	recurrence 0% $\rightarrow$ 100% (to 3,507 words); NLL +0.075

ends a sentence (real 0.6%). The topic state is the first count mechanism off the sharpening curve: better NLL than temperature sharpening at equal Distinct-2. Far agreement plus best-of-6 gives transformer-judged NLL 4.31–4.35 at Distinct-2 0.762–0.766. The transformer's own samples score 4.47 at 0.773. No sentence contains a copied 7-word span (transformer 0.7%), and none is a training sentence. **Caveats:** the judge is the transformer; our Distinct-2 is 0.007–0.011 lower; lower NLL can mean blander text (the transformer's own T=0.9 samples score 2.99 at 0.50). On the transformer's own metric we claim parity at near-matched diversity, not superiority. Uncurated samples are in Appendix F.

Human rating. Two raters, blind to source, rated 120 shuffled sentences on the same 1–3 scale as our earlier study: 40 each from this arm, the transformer's own T=1 samples, and real held-out text, each drawn by a fixed seed from 8–22-word sentences. The decision rule was fixed before rating. G* scored 2.31 (both raters gave 3 to 40% of items), the transformer 1.94 (10%), and real text 2.91 (92%). The difference is +0.375 (95% bootstrap interval [0.11, 0.63]; item-mean Mann–Whitney $p = 0.011$). Each rater separately ranked G* higher, with 86% exact agreement and weighted $\kappa = 0.80$. **Scope:** the transformer arm is its matched-diversity setting (Distinct-2 0.773 vs our 0.766). At its most fluent decoding (T=0.9, top-40; Distinct-2 0.50) an earlier study rated it 2.31, on different items. An earlier constraint-based version of our generator was rated 1.23 (Appendix G). The claim is therefore human preference at matched diversity, with $n = 40$ items per arm and two raters, and G* remains well below real text.

Referent recurrence at distance. In 100 trials of 256-sentence passages, the ledger brought the referent back in 100.0% of probes at every distance $d = 1, 2, 4, \ldots, 256$ sentences (about 14 to 3,507 words), with 1.00 construction attempts on average. G* without the ledger brought it back in 0% of probes at every distance, and so did the transformer (context 64 tokens); an earlier run gave 0–2% and 0–1%. The cost is small against ordinary G* sentences from the same passages: NLL 4.859 vs 4.784 (+0.075), D +0.166 vs +0.156, lengths within the real-text bars. A first version whose backward half lacked the states failed this quality bar (+0.41 NLL). Giving it the mirrored states fixed it.

6 The remaining gap lies beyond the window

The one axis on which the transformer is clearly ahead is D: +0.02 for our best text against -0.19 for real and transformer text. Much of the old D was run-on sentences (G* without an ending state: +0.39). What remains saturates. It sits at about +0.14 whatever the window (3, 4 or 5 words), bag or restriction strength. Far-agreement selection lowers it to about +0.03 at K=6 and then rises again as the selector over-optimises its proxy (+0.05 at K=24). Reading the text shows the same failure in most broken sentences: two locally fine halves collide at a seam ("he stooped and picked up / charlie jumped out"). Transformer errors are semantic, inside correct grammar.

Seventeen readers, none sees it. We tried to measure the collision with counts. We hand-labelled 200 generated sentences as broken or clean by reading them, before any scoring, in three separate draws. Each reader's pass bar was set in advance: detect collisions (AUC $\geq$ 0.65) or track D ($|\rho| \geq 0.15$). The readers were:

- **pair and class statistics:** exact-distance PMI, a class 8-gram D, clause-restart lifts;

- **induced grammars:** a Viterbi-EM arc grammar and a DMV with valence (Klein & Manning, 2004), whose inside chart was verified exact;
- **whole-prefix states:** class, mixed-alphabet and 20k-word HMMs fitted by Baum–Welch; a state fitted by CCA to predict the next 8 words; and a bilinear near×far version of it;
- **discriminative classifiers** separating real from generated text, local and whole-prefix;
- **a two-sentence test** (is this text more likely as two sentences glued at a seam?), also scored by a count model fitted on a disjoint half of the corpus;
- **a spectral weighted automaton** fitted by the method of moments (Hsu et al., 2012).

Only three of the seventeen track D at all: far agreement, the future state and the bilinear state (ρ =0.17–0.25). The bilinear state also remembers far context, which the Baum–Welch HMM does not (+0.267 nats against 0.000). None detects the collisions: all seventeen score 0.40–0.62 AUC, within chance or below the bar. Using any of them to steer or select lowers transformer-judged NLL but leaves D within noise of +0.02.

Why. Every count statistic evaluated on G*'s text is computed from the same counts G* assembled it from, so it rates G*'s seams as coherent by construction. Our text looks far *less* like two glued sentences than real text does (split log-odds -8.8 vs -1.8). Cross-fitting halves this bias: generating from one half of the corpus and scoring with the other gives -3.9 vs -1.6 and the best collision score (0.62). It does not remove the bias, because the halves share common phrases. Structurally, the collision is a clause that is already complete being restarted. Tracking it needs clause-level structure, the stack-shaped state that Corollary A.2 and Finding B place beyond bounded windows and flat states without a fixed convention. The transformer reads it with a gradient-fitted representation of its whole context, which our rules exclude. We also found that the transformer's D separates the hand labels only moderately (AUC 0.57–0.66), so part of the remaining D gap may be semantic rather than structural.

Where the gap lives. We decomposed the transformer's own long-context gain on clean real text (33,482 targets, $s(\text{last 5 words}) - s(\text{full prefix})$, mean 0.14 nats). Its last 8 words already hold 78% of the gain and its last 12 all of it, just beyond our 5-word window. At the level of our 48 count-derived word classes it gains almost nothing (+0.0015 nats), so the gain is in which word is chosen, not in these coarse classes. And 85% of the gain falls on words never seen after their last three words (81.5% of real words). G* continues the longest attested context, so only 17% of its words are of this kind and 83% were seen after their last 3–5 words. Every steer and selection above re-weights those well-attested words, where there is little long-context gain to have. This is a second, measurable reason why they saturate.

7 Discussion, limitations and conclusion

What we claim. On matched data and without gradients, three separate results. *Local prediction:* counting comes within 0.10 nats of the transformer wherever it has evidence, closes 78% of the likelihood gap, and loses the rest by composing unseen short contexts. The variable-length generator reaches the transformer's judged fluency at near-matched diversity, with zero copying at a competitor-calibrated budget. *Persistent identity:* exact referent recurrence at distances bounded context cannot reach, as Corollary A.1 predicts. *Clause coherence:* the remaining gap sits exactly where Corollary A.2 and Finding B say flat state stops. Throughout, per-token work is constant in corpus size. **What we do not claim:** human preference over every transformer decoding (only at matched diversity, in one small two-rater study), superiority on the transformer's own metric, or cross-sentence coherence, where the transformer is clearly ahead and seventeen count readers do not even see the defect. Also unmeasured: speed against the transformer on the same CPU, more than one corpus or language, and passages beyond single sentences apart from the ledger test. More data is not the route. Rebuilt from 1/8, 1/4, 1/2 and all of the corpus, the best generator's D is +0.062, +0.081, +0.021 and +0.027: -0.017 per doubling, and not monotone. Reaching -0.19 would take about 8,000x the data; the Gutenberg dump we used holds about 150x. Likelihood gains per doubling are shrinking too.

Conclusion. Each capability we measured corresponds to one kind of state. A bounded window gives local fluency, flat off-page state gives exact long-range identity, and clause-level coherence needs the stack-shaped state that neither provides. The compute cost of incorporating knowledge is

a property of gradient-trained generators, not of generation: a counting generator acquires knowledge in one pass and runs with constant process memory. The bounded-window ceiling is real and precisely located. Flat, off-page state escapes it for referent identity, and our generator achieves exact long-range recurrence that bounded context cannot. Within the window, counting reaches the transformer's judged fluency, and blind humans prefer its sentences to the transformer's own at matched diversity. The remaining gap, clause-level coherence, is where our theory says flat state stops. Closing it without gradients or conventions is the open problem this paper leaves, stated precisely rather than hidden.

Reproducibility Statement

All code is in the anonymized supplementary material. `python reproduce.py` recomputes every headline number in the paper from the shipped raw outputs in about a second and flags any mismatch; `python reproduce.py judge` re-scores the shipped generated sentences with the shipped transformer. There is one script per reported experiment, each with its pre-registered pass/fail rule in its header. The count tables are rebuilt from the public corpus (`manu/project_gutenberg` on Hugging Face) in one pass. The generator and the constant-work model run on a CPU; a GPU is used only to batch evaluation and to train and run the transformer baseline. Seeds and `PYTHONHASHSEED` are pinned. Hand labels are included with their sampling seeds. Verification was performed by the researchers and generative AI tools (Appendix A).

References

Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, 2000.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, 1995.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.

John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

Yangfeng Ji, Chenhao Tan, Sébastien Martschat, Yejin Choi, and Noah A. Smith. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 478–485, 2004.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 181–184, 1995.

Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*, 1998.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2016.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

Tomasz Marciniak and Michael Strube. Discrete optimization as an alternative to sequential processing in nlg. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG)*, 2005.

Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

Chris Mellish, Alistair Knott, Jon Oberlander, and Mick O'Donnell. Experiments using stochastic search for text planning. In *Proceedings of the Ninth International Workshop on Natural Language Generation (INLGW)*, 1998.

William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

John Myhill. Finite automata and the representation of events. *WADC Technical Report 57-624*, 1957.

Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9:541–544, 1958.

Richard Power. Planning texts by constraint satisfaction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, 2000.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10(3):187–228, 1996.

Lawrence Saul and Fernando Pereira. Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 81–89, 1997.

Sebastian Varges and Chris Mellish. Instance-based natural language generation. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2001.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

A AI USE STATEMENT

In this work, we used generative AI tools (Claude, Anthropic) for: developing the theoretical framing and formal claims in Section 3, including formulating and proving Theorem A, Corollary A.1, and Corollary A.2; designing the research methodology and experiments throughout Sections 4–6; implementing the code for our methods, experiments, and analysis; cleaning, reformatting, and building acquisition tables from the underlying corpus; labelling generated sentences as broken or clean by reading them (Section 6; labels fixed before any scoring); and interpreting experimental results, including root-causing negative and null results reported in this paper. Earlier naturalness ratings of the previous generator used both zero-context AI raters and a double-blind study with two human raters (Appendix G). For the current generator, fluency and coherence numbers are computed by the converged transformer baseline, and naturalness was rated by two human raters blind to source; no AI rater was used. Reproducibility verification was performed both by the researchers directly and by generative AI tools running the majority of these checks. We did not use generative AI tools to generate synthetic data sets — all reported results are measured on real, publicly available Project Gutenberg text and real trained-model outputs. Additionally, we used generative AI tools (Claude, Anthropic, and, for gathering an initial candidate list of related-work citations, one additional generative AI tool) for: creating the tables and figures; suggesting experimental parameters; producing the anonymized supplementary-material artifact; drafting a substantial majority of this manuscript's prose, including the abstract; summarizing and analyzing existing literature; searching for and formatting references; and suggesting this paper's structure. The researchers set the research direction and priorities throughout this project — what to investigate, when a result was sufficient to report, and when to conclude an unproductive line of investigation — and independently reviewed, verified, and take full responsibility for all content generated with AI assistance. Every experimental result was run and its output checked before inclusion, including negative and null results; every citation proposed by a generative AI tool was verified before use; every mathematical claim and proof was reviewed by the researchers. During this review an error in our own reference coherence metric was found and corrected (Appendix E). We take responsibility for the final content of this work.

B FULL AUTOMATON CONSTRUCTION FOR THEOREM A

Let M be the automaton whose state after reading $w_1 \dots w_j$ is the length-$\min(2k{+}1, j)$ suffix of tokens read so far, padding on the left with a distinguished boundary symbol $\perp \notin \Sigma$ for the first $2k$ positions, so the buffer always represents the most recent $2k{+}1$ read positions once $j \geq 2k{+}1$. For $j \leq 2k$ the padding amount is determined by j alone, so there are exactly $|\Sigma|^j$ such states at that j; for $j \geq 2k+1$ the buffer ranges over Σ^{2k+1}. The number of states is thus bounded by $\sum_{t=0}^{2k+1} |\Sigma|^t$ — finite, and independent of n. On reading token w_{j+1}, M shifts its buffer by one and, once the buffer is centered on position $j+1-k$ (i.e. it holds $w_{j+1-2k}, \dots, w_{j+1}$, giving the full $2k{+}1$-token neighborhood of that position), evaluates A_k there, transitioning to a dead-reject state if it fails. The final k positions (within k of the string's right end) lack real right-context once the last token is read; their evaluation is folded into M's end-of-string accepting-state computation, using the stored buffer with missing right-context treated as $\perp$ — a function of M's final state alone, not an operation on

further input. M accepts iff every position was evaluated this way and none failed. By construction, M accepts exactly $L(A_k)$, and M's state count is independent of n; hence $L(A_k)$ is regular. ■

What the ledger escapes. The fixed-radius (locally testable) class, not regularity as such. Over a fixed finite vocabulary the ledger's state has at most $2^{|\Sigma|}$ values, so ledger-plus-window is itself finite-state; it adds recurrence at *unbounded distance* with state independent of that distance, which no fixed-radius predicate supplies. We do not claim it recognizes ordered non-regular languages such as the copy language.

C Full proof of Corollary A.1

Setup. Let $\Sigma_0 \subset \Sigma$ with $|\Sigma_0| \geq 2$ be a sub-alphabet used to name referent identities, and let $\# \in \Sigma \setminus \Sigma_0$ be a boundary token used only inside this proof's formal abstraction, never by the real generator (the referent ledger has no reserved marker anywhere in its pipeline). For $u = u_1 \cdots u_m \in \Sigma_0^m$, let $w_u = u_1 \cdots u_m \# u_1 \cdots u_m$, modeling a passage whose first half introduces m referents and whose second half must reproduce them in order. Let $L_{copy} = \{v\#v : v \in \Sigma_0^*\}$.

Lemma. L_{copy} is not regular.

Proof. Suppose L_{copy} is regular, with pumping length p. Take $w = 0^p \# 0^p$. The lemma gives $w = xyz$ with $|xy| \leq p$, $|y| \geq 1$, $xy^\ell z \in L_{copy}$ for all $\ell \geq 0$. Since the first p symbols are 0^p, $y = 0^t$ with $1 \leq t \leq p$, and $xy^0z = 0^{p-t} \# 0^p$ has halves of unequal length, so it is not in L_{copy}. Contradiction. ■

Proof of Corollary A.1. Suppose some A_k of finite radius enforced exact recurrence, in the sense that wherever $\Sigma_0^*\#\Sigma_0^*$ occurs the two halves must match. Then $L(A_k) \cap (\Sigma_0^*\#\Sigma_0^*) = L_{copy}$. $L(A_k)$ is regular by Theorem A and $\Sigma_0^*\#\Sigma_0^*$ is regular, so the intersection is regular, contradicting the Lemma. ■ The $\#$ token and two-symbol Σ_0 are devices of the reduction only; they correspond to nothing in the marker-free generator.

D Corollary A.2's target, restated formally

Fix a finite noun set N and a finite past-tense transitive-verb set V. Let $R = (\text{the} \cdot N)^+ \cdot V^+$, regular. Let L be the (idealized, competence-level) set of grammatical English sentences. Within R, grammaticality requires the noun count to equal the verb count (n nested subjects need exactly n verbs, closed last-opened-first-closed), so $L \cap R = \{(\text{the}\, w_1) \cdots (\text{the}\, w_n)\, v_1 \cdots v_n\}$. The homomorphism h erasing "the" and mapping $N \mapsto a$, $V \mapsto b$ gives $h(L \cap R) = \{a^n b^n : n \geq 1\}$, not regular by the pumping argument of Appendix C ($w = a^p b^p$). If L were regular, $h(L \cap R)$ would be regular by closure under intersection with a regular language and under homomorphism. Contradiction; hence L is not regular, and by Theorem A no finite-radius A_k is sound and complete for it at unbounded depth (Chomsky, 1956; Hopcroft & Ullman, 1979).

E Coherence readers: full list, and a correction

Each reader was pre-registered with two bars: collision AUC ≥ 0.65 on the hand labels (broken sentences scored as less coherent), and $|\rho| \geq 0.15$ with the transformer's per-sentence D on 6,000 generated candidates. Labels: 40 + 80 sentences drawn from the full-corpus best-of-6 arm, and 80 from the cross-fitted generator, each set drawn by a fixed seed and labelled by reading before scoring. Results (collision AUC; ρ): far agreement ($\rho = -0.21$, qualifies); exact-distance PMI (no; $\rho \leq 0.11$); class 8-gram D (no); clause-restart lifts (no); Viterbi-EM arc grammar (collapsed to a chain, no); DMV with valence, exact inside chart, stochastic EM (0.50–0.51; no); class HMM (Baum–Welch, 128 states; 0.32–0.45); mixed-alphabet HMM (256 states; 0.29–0.46); word HMM (1,024 states; forgets within 5 words, $D \equiv 0$); future-predicting state (CCA, rank 256; memory +0.196, $\rho = +0.17$, collision 0.40); bilinear near×far state (memory +0.267, $\rho = +0.20$, collision 0.42); discriminative, local n-gram features (real-vs-ours AUC 0.68, collision 0.50); discriminative, whole-prefix state features (0.66, 0.54); two-sentence split test (0.58); the same under a grammar (0.51); cross-fitted two-sentence test (0.62, $\rho = +0.04$); spectral automaton, rank 64 (memory

+0.028, $\rho = -0.01$, collision 0.45). All readers scored below chance on the first 40 labels. That pattern did not replicate on the 80 fresh labels (0.48–0.55).

Correction. The transformer "reference" D we first computed on the labelled sentences averaged over end-of-sentence and unknown-word targets as well. The judge's D (used in every table and every ρ bar) excludes both. With the judge's D, the transformer separates the labels at 0.57, 0.66 and 0.57 on the three draws, not the 0.72–0.76 our first computation gave. One cross-fitted reader's ρ fell from +0.25 to +0.04 under the correct D and is reported as failing.

F UNCURATED SAMPLES

Drawn by a fixed seed (2027), not selected. The first six are from the best-of-6 far-agreement arm of Table 2 (lower-cased, punctuation stripped as in the corpus); the last four are the transformer's own T=1 samples. The corpus's licence boilerplate and dialect spellings appear in both.

G*	but we dunnot think as mich talk when they're getten theer
G*	hartley sank back in her chair exhausted with the violent exertions
G*	nell and drake withdrew to the table and the chairs
G*	for he did not want any more bother with them
G*	end of the project gutenberg ebook is public domain in the public
G*	the region is famous besides for the enterprise
TF	we refer to a named age who owe great delight to the previous day etc
TF	what did you in it to me is that the thing you show
TF	into the apartment he left along the scow which had been sleeping in the straw in the meantime
TF	bei se h n l en en m y nyt ei ei on n s me ja thar ane pe n

G THE EARLIER CONSTRAINT-BASED GENERATOR

Before G*, clauses were built by AC-3 arc-consistency (Mackworth, 1977). Each of 9 word positions was a variable whose domain was pruned to words satisfying bigram and skip-gram attestation at offsets up to ± 5 (the A_k, k=5, of Theorem A), with weighted-random choice among admissible values. The ledger forced a tracked referent into a passage's final clause. **Human study:** two humans, double-blind, rated 160 items (1–3 scale; 40 each). The constraint generator scored 1.225, an unconverged 12-minute transformer 1.113, the converged transformer 2.312 and real text 2.812; weighted $\kappa = 0.88$, $p = 8 \times 10^{-13}$ for the converged transformer against the constraint generator. Its clauses were 9 words long and the transformer's about 14, so raters could partly tell arms apart by length. An earlier LLM-rater preference ($z = +2.9$) and a diversity advantage (Distinct-2 0.91 vs 0.79) held only against the unconverged transformer and are withdrawn. **Acquisition:** building its five tables from 10K to 1.22M sentences took 63–756 s ($\sim N^{0.51}$) on a CPU. Memory was not flat (peak 190 MB to 3.3 GB), correcting an earlier figure that timed only GPU clustering while loading the full corpus at every size (Figure 2). A larger corpus made its constraint search 3.4–4.8x faster per passage over two seeds. **External-model perplexity:** under GPT-2-small its clauses scored perplexity 634 and its passages 449, a disclosed loss. Perplexity rewards generic text (Holtzman et al., 2020), and this generator was biased toward rare, specific words. Its referent pool did not saturate to 1,500 passages (Figure 3). Six pre-registered local fixes for its fluency all failed. The same clustering applied to GPT-2-small's literal embedding matrix tied or exceeded our from-scratch classes on 3 held-out categories.

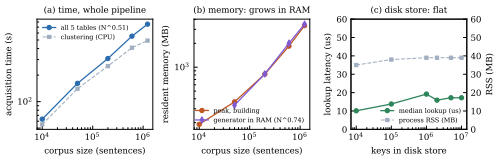


Figure 2: Acquisition of the earlier generator, whole pipeline, CPU.

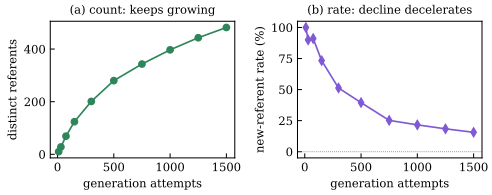


Figure 3: Referent-pool growth to 1,500 passages.