Likelihood Gains Do Not Establish Long-Range Memory Use: Stress-Testing Compressed Plan Memory with Plan-Bank Interventions

Anonymous ACL submission

Abstract

Augmenting a language model with a learned compressed memory often lowers held-out likelihood, and such gains are routinely read as evidence that the model uses the memory to track long-range state. We argue that this inference is unsafe, and we make it testable. We study PLANGEN, a decoder-only model with an explicit plan bank over token blocks whose vectors we can directly corrupt by swapping, scrambling, dropping, or misaligning them. This gives a paired causal test on long-range positions rather than an aggregate likelihood comparison. On an easy templated source the plan channel produces large gains of up to a 76.4% NLL reduction in a non-causal diagnostic, with strong responses to every corruption. On a richer source with genuine beyond-window dependencies, the same architecture improves average likelihood but shows no confirmed advantage on the long-range positions the memory should govern; there, removing plan information still hurts, but corrupting its order or alignment does not, so the memory carries information without being used in a position-aligned way.

1 Introduction

Long-range language modeling requires more than fitting a long sequence into a model's context window. Benchmarks for long-context language models show that extended context length alone does not guarantee reliable retrieval, aggregation, or reasoning when evidence is distributed across long inputs or encoded in latent structure (Bai et al., 2024b; Hsieh et al., 2024; Vodrahalli et al., 2024; Modarressi et al., 2025). This issue is central to narrative and document modeling, where a model must preserve entity attributes, discourse commitments, latent goals, and event order across spans that may exceed a bounded local context.

One influential response is to give the model a learned, compressed memory: a small set of vectors that summarize earlier text and that the model conditions on while predicting later tokens (Dai et al., 2019; Rae et al., 2020; Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Hutchins et al., 2022). Such memories frequently lower held-out likelihood, and these gains are commonly taken as evidence that the model is using the memory to carry long-range state. We argue that this reading is unsafe. A compressed channel can lower average likelihood for reasons unrelated to long-range use, including added capacity or easier modeling of nearby tokens, so an aggregate likelihood improvement does not by itself show that the memory is read where long-range state actually matters.

We therefore study compressed memory through direct intervention rather than likelihood alone. Different corruptions probe different things: dropping the memory tests whether the channel carries useful information at all, while scrambling its order or misaligning it with the text tests whether the model uses the memory in a position-sensitive way. If the memory genuinely tracks which long-range state belongs at which later position, then order and alignment corruptions should hurt precisely at the long-range positions; if average likelihood improves yet these corruptions leave long-range positions unchanged, the gain is not evidence of position-aligned long-range use. This turns a vague claim about memory use into paired, falsifiable tests.

As a concrete instantiation, we study PLANGEN, a decoder-only language model with an explicit plan bank over fixed-size token blocks.[1] A plan encoder maps each block to a compact plan vector; the resulting vectors form a plan bank that the decoder reads while predicting tokens. PLANGEN is a deliberately inspectable instance of the compressed-

[1]The generator, the generated synthetic documents and their gold metadata, and the intervention, future-leakage check, and evaluation code are available at https://anonymous.4open.science/r/plangen.

memory designs above: its block-aligned memory is a clean object to corrupt, so we can vary its capacity, restrict which tokens may determine each plan vector, and directly swap, scramble, drop, or misalign its entries.

We apply this framework across two controlled sources with different long-range structure, and the picture changes sharply. On an easy, heavily templated source the plan channel behaves as the optimistic reading predicts: a capacity diagnostic cuts held-out NLL by 76.4% and all four corruptions increase loss, which taken alone looks like strong evidence of causal long-range use. That evidence does not carry over to a harder source whose dependencies span beyond the local window. There the same architecture still lowers *average* NLL, but its advantage on the long-range positions the memory should govern is not statistically confirmed: removing plan information wholesale still hurts, yet corrupting its order or alignment does not. The memory carries useful information without being used in a position-aligned way for long-range structure. Aggregate likelihood, the metric usually reported for such memories, is precisely what fails to reveal this.

Our contributions are: (1) an intervention-based framework for testing whether a learned compressed memory is causally used for long-range modeling, which corrupts an explicit block-aligned plan bank by swapping, scrambling, dropping, and shifting its vectors and measures the effect specifically on long-range positions under paired evaluation; (2) PLANGEN, a representative decoder-only instantiation, with access restrictions and an equal-parameter control that isolate capacity, non-causal information, and causal use; (3) evidence that on an easy templated source the framework reports strong causal use, including a 76.4% capacity-diagnostic NLL reduction and uniformly damaging corruptions; and (4) the contrasting finding that on a richer source with genuine long-range dependencies the same architecture improves only average likelihood, with no confirmed long-range advantage and order or alignment corruptions that no longer bite even though wholesale removal still does, showing that likelihood gains do not establish position-aligned long-range use of the memory.

2 Related Work

Learned memory and context compression. Closest to our work are methods that give a Transformer a learned internal memory. Some add recurrent or compressed state through segment-level recurrence, compressed memories, memorized key-value stores, or recurrent memory tokens (Dai et al., 2019; Rae et al., 2020; Wu et al., 2022; Bulatov et al., 2022); others compress spans of text into compact continuous vectors the model later conditions on, including prompt gisting (Mu et al., 2023), summary-vector autocompression (Chevalier et al., 2023), in-context autoencoding into memory slots (Ge et al., 2024), and block-recurrent state propagated through gated fusion (Hutchins et al., 2022). These methods share PLANGEN's use of a learned compact memory derived from text, and some adopt comparable gating, but they target efficiency or context extension and are evaluated by downstream quality or perplexity. That evaluation is exactly what we question: a likelihood improvement is routinely read as evidence that the memory is used, yet it does not show the memory is read where long-range state matters. We therefore treat the compressed memory as an explicit object of causal study, corrupting it through paired interventions to test whether the decoder relies on its content, order, and alignment, rather than only whether it lowers likelihood.

Long-context, retrieval, and planning. Three further lines of work address long-range modeling differently. Benchmarks for long-context models show that nominal context length does not guarantee robust use of distributed or latent-structured evidence (Bai et al., 2024b, 2025; An et al., 2024; Yuan et al., 2024; Liu et al., 2024b; Li et al., 2024; Hsieh et al., 2024; Yen et al., 2024; Kuratov et al., 2024; Vodrahalli et al., 2024; Modarressi et al., 2025); context-extension and efficient-attention methods enlarge or cheapen the usable window (Peng et al., 2024; Chen et al., 2024; Ding et al., 2024; Bai et al., 2024a; Ding et al., 2023; Liu et al., 2024a; Xiao et al., 2024b,a; Reid et al., 2024); and retrieval-augmented systems supply non-parametric memory as external objects selected at inference time (Asai et al., 2024; Sarthi et al., 2024; Edge et al., 2024; Guo et al., 2024; Jiménez Gutiérrez et al., 2024, 2025; Qian et al., 2024). Planning-based generators add explicit structure as outlines, pre-writing, or agentic scaffolds around a frozen generator (Lee et al., 2025; Xie and Riedl, 2024; Shao et al., 2024; Wang et al., 2024; Li et al., 2025; Xia et al., 2025; Xu et al., 2025). None of these makes a model's internal dis-

course state an explicit, directly corruptible object: a longer window or a retrieved passage exposes more tokens but does not provide a memory whose content, order, and alignment can be intervened on, which is the test our framework requires.

3 Method

We study whether explicit block-level memory improves long-range conditional language modeling, and whether any improvement reflects genuine use rather than added capacity or non-causal access. We pair a controlled synthetic source, which provides known latent structure and delayed dependencies, with an explicit plan bank that can be corrupted directly. A parameter-matched control isolates capacity, leakage-free access restrictions remove non-causal information, and direct plan-bank interventions test whether the memory is causally read. Given a token sequence $x_{1:T}$, we divide it into fixed-size contiguous token blocks of size r and let

$$b(t) = \left\lfloor \frac{t-1}{r} \right\rfloor \tag{1}$$

denote the block containing position t. A local-window baseline predicts the next token from a bounded history,

$$Q_\theta^{\text{local}}(x_{t+1} \mid x_{t-k+1:t}), \tag{2}$$

where k is the local context window. PLANGEN augments this decoder with a compact plan bank:

$$Q_\theta^{\text{plan}}(x_{t+1} \mid x_{t-k+1:t}, \hat{P}_{\leq b(t)}), \tag{3}$$

where $\hat{P}_{\leq i}$ denotes the plan memory available up to block i, and $b(t)$ is the block containing position t, so the decoder reads $\hat{P}_{\leq b(t)}$ when predicting x_{t+1}.

Block-level plan bank. A plan encoder maps block representations to compact plan vectors. Given decoder hidden states $h_{1:T}$, we pool hidden states within each block and project them through a plan encoder:

$$\begin{aligned} \bar{h}_i &= \text{Pool}(\{h_t : b(t) = i\}), \\ \hat{p}_i &= f_\phi(\bar{h}_i), \\ \hat{p}_i &\in \mathbb{R}^{d_p}. \end{aligned} \tag{4}$$

The resulting plan bank and its prefix are

$$\hat{P} = [\hat{p}_0, \hat{p}_1, \ldots, \hat{p}_{B-1}], \qquad \hat{P}_{\leq i} = [\hat{p}_0, \ldots, \hat{p}_i], \tag{5}$$

where $B = \lceil T/r \rceil$. At position t, the decoder reads the plan available for the current block:

$$r_t = \text{Read}(\hat{P}_{\leq b(t)}, b(t)) = \hat{p}_{b(t)}. \tag{6}$$

The read vector is projected as $z_t = W_p r_t$ and fused with the local hidden state through a gate, $\tilde{h}_t = h_t + g_t \odot z_t$ with $g_t = \sigma(W_g[h_t; z_t])$. The language-model head predicts from $\tilde{h}_t$. Figure 1 summarizes the architecture: the token stream is divided into blocks, the plan encoder writes compact vectors into a plan bank, and the decoder reads the plan memory alongside local token history.

Causally restricted plan access. A causally restricted plan source computes every available plan only from earlier blocks, never from the block being predicted or a later one, unlike the non-causal block-hidden diagnostic that may use the current block under teacher forcing. We evaluate previous-block and prefix-online variants, and verify the absence of future-token leakage by checking that changing future tokens does not alter earlier plans.

$$\hat{p}_i = \begin{cases} p_{\text{BOS}}, & i = 0, \\ f_\phi(\bar{h}_{i-1}), & i > 0. \end{cases} \tag{7}$$

In the prefix-online variant, rather than assigning one plan per block, the plan available at position t is computed only from the prefix $x_{1:t}$ seen so far, so it updates as the prefix grows and never depends on tokens at or after t.

Plan-bank interventions. Because the plan bank is explicit, we corrupt it directly and measure the paired change

$$\Delta_{\text{NLL}} = \text{NLL}_\theta(x_{1:T}; \hat{P}') - \text{NLL}_\theta(x_{1:T}; \hat{P}), \tag{8}$$

where $\hat{P}'$ is a corrupted plan bank and both terms score the same held-out sequence, so the only changed object is the plan bank. A positive Δ_{NLL} means corruption worsens prediction. We use four paired interventions: swapping plans across documents, scrambling plan order, dropping plan vectors, and misaligning plans with token blocks.

Controlled continuation ranking. We also test whether plan memory helps rank state-consistent continuations above corrupted alternatives. For each prompt, we construct a gold continuation and corrupted candidates, score each with teacher-forced NLL, and report whether the gold continuation is ranked first. This controlled continuation-ranking task evaluates long-range state consistency without relying on open-ended generation.

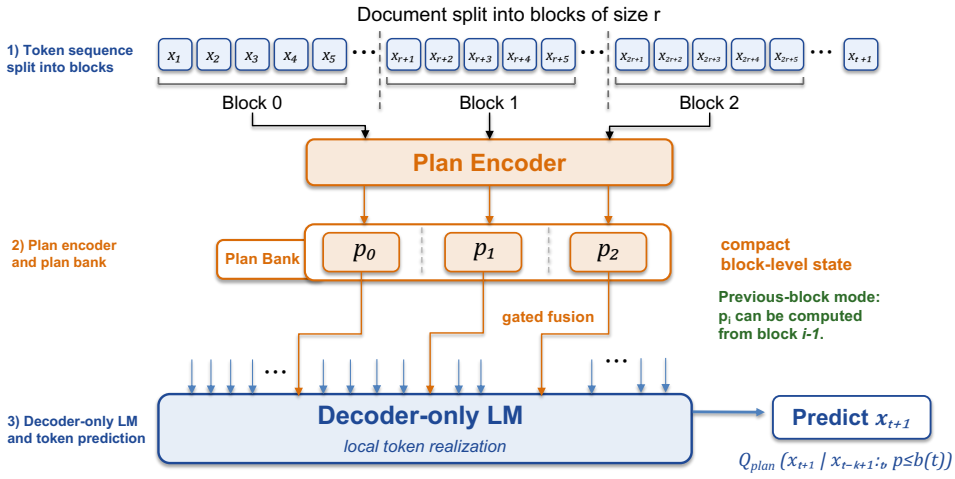

Figure 1: PLANGEN augments a decoder-only language model with explicit memory over fixed-size token blocks. A plan encoder maps each block representation to a compact plan vector; the resulting vectors form a plan bank read by the decoder through gated fusion. This separates local token realization from longer-range state tracking and makes the memory channel directly intervenable.

Additional implementation details. We focus here on the core plan-bank mechanism; Appendix B gives additional method details, including the local-window attention mask, parameter-matched local baselines, alternative fusion choices, noisy and quantized plan channels, full plan-source modes, formal intervention definitions, and the continuation-ranking metric.

4 Experimental Setup

We evaluate PLANGEN on two controlled sources of differing difficulty, through held-out likelihood, direct plan-bank corruptions, future-leakage-checked previous-block confirmation, and controlled continuation ranking. The easy source supports our capacity and corruption analyses; the harder source, with genuine beyond-window dependencies, tests whether the apparent causal use generalizes.

Controlled synthetic benchmark. Our main benchmark is a synthetic narrative source designed to expose long-range structure that cannot be recovered from a bounded local window alone. Each document is generated from a global genre variable, an entity table, and a sequence of block-level latent states. The generator lexicalizes these states into text blocks while recording gold metadata for latent state, entity tuples, coreference links, and contradiction spans. This gives us a setting where persistent document state is known by construction, while surface forms can vary across blocks.

The source is designed to make plan-memory interventions interpretable. If the decoder ignores the plan bank, swapping, scrambling, dropping, or boundary-shifting plan vectors should have little effect. If the decoder uses the plan bank, these corruptions should increase held-out NLL. We use this benchmark for our capacity diagnostic, corruption tests, and previous-block confirmation. We refer to it as the easy source: its vocabulary is small, its templates are few, and its dependencies are often restated within the local window, so a bounded model can frequently resolve them. To support scrutiny of the source design, we release the generator, the generated documents, and the metadata it records together with our evaluation code.

Harder long-range source. Because the easy source can overstate how much a memory is used, we construct a second controlled source designed to make long-range use necessary and measurable. It draws entities, attributes, and values from much larger pools, varies templates, and enforces dependencies whose governing value is established far earlier and not restated within the local window before it is needed. We record, for each dependency, the distance to its governing value and whether it is locally restated, letting us identify and separately score the long-range positions where the relevant evidence lies outside the local window. This source is the setting for Section 5.1.

Controlled source design. The controlled source is a narrative-HMM generator with explicit document state: each document samples a genre, an

Component	Role in source	Evaluation use
Genre G	Defines document-level style and transition regime	Creates global context shared across blocks
Entity table E	Stores entities, attributes, roles, and goals	Supports entity swaps and consistency checks
Latent state z_i	Controls block-level event progression	Tests whether memory tracks delayed state
Surface variation	Adds paraphrases and filler around latent events	Prevents exact template matching
Contradiction templates	Violate earlier entity or event commitments	Construct corrupted continuation candidates
Held-out splits	Separate train, validation, and test documents	Enable validation-only selection and final testing

Table 1: Controlled source components and how they support the evaluation. The source separates local surface realization from persistent document state, enabling direct tests of plan memory.

entity table, and a block-level latent state sequence, which are lexicalized into block text. Blocks maintain entity attributes, continue event threads, refer back to earlier entities, or introduce controlled contradictions, and the generator records gold metadata for latent states, entity tuples, coreference links, and contradiction spans. These labels do not supervise the model; they are used to build controlled corruptions and continuation-ranking candidates. The design separates local surface realization from longer-range state: a bounded model sees recent tokens but may miss commitments established several blocks earlier, whereas a plan-memory model can compress that state into plan vectors. Full generator equations are in Appendix C.

Table 1 summarizes the source components and their role in evaluation. The key property is that the generator exposes persistent state by construction while still allowing local surface variation. This lets us test whether plan memory captures block-level state rather than merely exploiting repeated templates. Appendix C provides the full generator equations, entity and event metadata, surface-variation details, Gaussian diagnostic source, and split/leakage checks.

Models and selection. We compare local-window baselines against plan-memory models that vary plan dimension d_p, block size r, and plan access mode. Non-causal block-hidden models can compute plans from the current block under teacher forcing. Causally restricted variants use either previous-block or prefix-online plan access (Section 3). For confirmatory runs, we select models using validation metrics and evaluate the selected configuration once on held-out test data. We additionally use a parameter-matched local adapter, an equal-parameter local model with extra non-plan parameters but no block-level plan bank, as a capacity control that tests whether plan-model gains are explained by parameter count alone. Full model and training settings for every reported run are in Appendix A.

Metrics. The primary metric is token-level held-out negative log-likelihood. For paired model comparisons, we report local-minus-plan NLL:

$$\Delta_{\text{local}-\text{plan}} = \text{NLL}_{\text{local}} - \text{NLL}_{\text{plan}}. \qquad (9)$$

Positive values indicate lower NLL for the plan model. For plan-bank interventions, we use the paired corruption delta in Eq. 8; positive values indicate that corrupting memory hurts prediction.

Selection protocol. For our confirmatory results, we follow a strict validation-only selection protocol: every choice of model, hyperparameter, and (where relevant) intervention is made using validation data alone, and the held-out test set is evaluated only once, after all such choices are fixed. Candidate plan models are selected by validation NLL, and for intervention-sensitive claims also by positive validation intervention deltas. The test set is never used for model selection, decoding choices, or intervention tuning, so the reported test numbers are not optimized against. We use this protocol for the confirmatory comparisons reported below. Separately, for exploratory analyses we simply report the full grid of trained configurations rather than selecting among them.

Causally restricted and continuation-ranking evaluations. For causally restricted confirmation, the plan available to a block may not depend on that block or any later one, which we enforce with future-token invariance checks (Section 3); the selected previous-block model is evaluated once against a local-window baseline on held-out documents. For controlled continuation ranking, each prompt is paired with a gold continuation and corrupted alternatives, models score all candidates with teacher-forced NLL, and we report whether the gold continuation is ranked first.

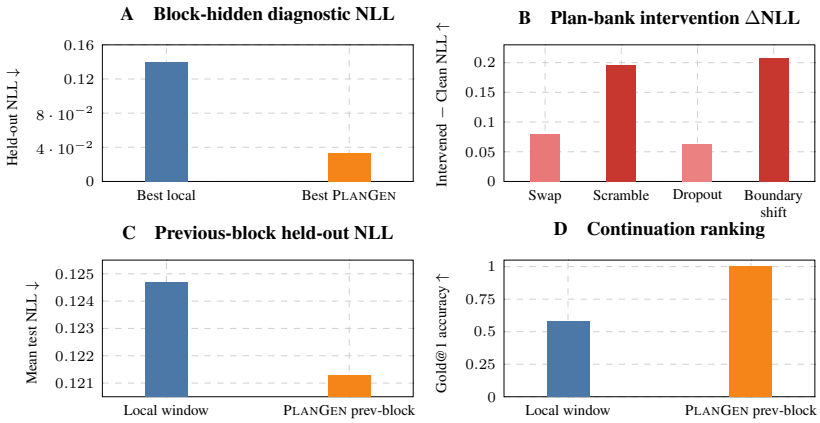

Figure 2: Easy-source evidence, all pointing to apparent causal use. A: a non-causal block-hidden diagnostic reduces held-out NLL $0.140 \rightarrow 0.033$ (76.4%). B: all four plan-bank corruptions increase NLL. C: the future-leakage-checked previous-block model improves over a local-window baseline ($+0.003$; CI above zero; 8/8 seeds; zoomed). D: it ranks state-consistent continuations above corrupted ones far more reliably (Gold@1 1.000 vs. 0.583).

5 Results and Analysis

We first present results on an easy, heavily templated controlled source, where the plan channel appears to be strongly and causally used. We then apply the same framework to a harder source with genuine beyond-window dependencies, where that appearance does not carry over. The contrast between the two is our main finding.

For the easy source we report held-out likelihood, plan-bank corruptions, corruption-severity analysis, and a future-leakage-checked previous-block confirmation. The block-hidden diagnostic measures how much latent state a block-level plan channel can carry when plans may be computed from the same block under teacher forcing, while the previous-block setting tests whether plan memory helps without current-block or future-token access.

Block-hidden plan-capacity diagnostic. Figure 2A shows the non-causal block-hidden diagnostic result. The best local baseline obtains NLL 0.139646, while the best block-hidden plan-memory model obtains NLL 0.032944, a 76.4% relative reduction. This result should be interpreted as a plan-capacity diagnostic: when compact plan vectors can be constructed from same-block hidden states under teacher forcing, they carry substantial information about the controlled latent state.

The improvement is not explained by local block size alone: both local baselines remain near 0.14 NLL, while multiple plan variants improve substantially, the strongest combining a 256-dimensional plan vector with block size 32. The full grid is in Appendix C.7.

Plan-bank interventions. Figure 2B shows that all four controlled plan-bank corruptions increase NLL. Swap, scramble, dropout, and boundary shift yield mean ΔNLL values of $+0.079727$, $+0.195824$, $+0.062496$, and $+0.208242$, respectively. Scramble and boundary shift are the most damaging interventions. Scrambling preserves the set of plan vectors but destroys their order, while boundary shift preserves local order but misaligns plans with token blocks. The degradation under both perturbations indicates that the decoder relies not only on plan-vector content, but also on temporal alignment between plan memory and token blocks.

Corruption severity. We further test whether stronger plan corruption produces stronger degradation. Across the tested severity grid, dropout, boundary shift, and scramble produce monotonic increases in NLL degradation, with zero severity yielding zero delta. This monotonic pattern strengthens the causal-use interpretation: increasingly severe corruption of the plan channel produces increasingly severe likelihood degradation.

Future-leakage-checked previous-block confirmation. The non-causal block-hidden diagnostic can compute plans from the current block under teacher forcing, so we additionally evaluate a previous-block model that is checked for future-token leakage. Figure 2C compares the trained local-window baseline with the previous-block PLANGEN variant chosen by validation NLL on held-out test data. The plan model improves mean

test NLL from 0.124674 to 0.121282, with local-minus-plan difference $+0.003393$; the paired bootstrap confidence interval is above zero and the plan model wins all eight seeds. Because the plan for block i is computed from block $i-1$, this result rules out current-block access as the explanation for the gain. To test whether the improvement is explained by added capacity, we also evaluate a parameter-matched local adapter on the same held-out test documents, using the same selection protocol, tokenizer, split, and training budget. The parameter-matched local adapter improves only slightly over the plain local-window baseline (NLL 0.124502 vs. 0.124674), whereas the causally restricted plan model reaches NLL 0.121282. Thus, under the matched-control setting, the plan model retains a $+0.003220$ NLL advantage over an equal-parameter local adapter. We interpret this result as a stringent future-leakage-checked confirmation in a low-NLL controlled regime, rather than as the main estimate of effect size.

Controlled continuation ranking. Figure 2D evaluates controlled synthetic continuation ranking, where each prompt is paired with a gold continuation and corrupted alternatives. The local-window baseline reaches Gold@1 accuracy 0.583, while the previous-block plan model reaches 1.000. The plan model also yields a larger NLL margin between corrupted and gold continuations.

The easy source looks like strong causal use. Taken together, the easy-source results form a coherent and seemingly strong case: a large capacity-diagnostic gain, a leakage-checked previous-block improvement that survives an equal-parameter control, four corruptions that all increase loss, graded sensitivity to corruption severity, and near-perfect preference for state-consistent continuations. Read in isolation, this is exactly the evidence one would cite to claim that the decoder causally uses plan memory to track long-range state. The next section shows that this conclusion is premature.

5.1 A Harder Source Removes the Position-Aligned Signal

The easy source is heavily templated, with a small vocabulary and dependencies that are often restated within the local window. To test whether the apparent position-aligned use generalizes, we build a second controlled source with a large vocabulary, varied templates, and dependencies whose governing value is established far earlier and not restated within the local window before it is needed, so a local-window model lacks direct access to it and any genuine long-range use should appear there. An audit confirms its long-range character: at the most long-range setting the governing value is outside the local window for 99% of dependencies, a local-window model resolves only about 12% of long-range positions, and we evaluate 825 such positions across held-out documents. We compare PLANGEN against local-window and equal-parameter local baselines under the same validation-only protocol, reporting NLL overall and on those long-range positions.

Comparison	Overall ΔNLL	Long-range ΔNLL
vs. equal-param. local	$+0.017$	-0.026
95% CI	$[+0.011, +0.024]$	$[-0.115, +0.071]$
vs. local-window	$+0.036$	-0.067
95% CI	$[+0.026, +0.046]$	$[-0.091, -0.043]$

Table 2: Harder source, most long-range setting (non-local fraction 0.99; $n = 5$ seeds, 825 long-range positions; paired bootstrap CIs). ΔNLL is baseline minus PLANGEN; positive favors PLANGEN. The overall gain is confirmed; on long-range positions the gain over the equal-parameter control is not confirmed and PLANGEN is worse than the local-window baseline.

Average gains persist, long-range gains do not. On the harder source the same architecture lowers overall held-out NLL over both baselines, winning all five seeds against the local-window baseline. But the picture inverts on long-range positions. Table 2 contrasts the two: the overall advantage over the equal-parameter control is confirmed, whereas on long-range positions PLANGEN has no confirmed advantage at all. Against the equal-parameter control the long-range interval includes zero, and against the local-window baseline PLANGEN is in fact worse on long-range positions. The gain that survives is an average-fit gain, not a position-aligned long-range one.

Order and alignment corruptions stop biting. The intervention profile makes the same point more sharply. Table 3 compares long-range corruption deltas on the two sources. On the easy source, scramble and boundary shift, which preserve plan content but destroy order or alignment, were the most damaging corruptions. On the harder source their effect on long-range positions collapses to near zero, while only dropout, which removes plan information wholesale, still degrades prediction. A memory that genuinely encoded which long-range

Corruption	Easy	Harder long-range
Swap	$+0.080$	$+0.0002$
Scramble	$+0.196$	$+0.0005$ $[-0.0003, +0.0012]$
Dropout	$+0.062$	$+0.510$
Boundary shift	$+0.208$	-0.0005 $[-0.0020, +0.0011]$

Table 3: Corruption ΔNLL on the easy source versus harder-source long-range positions ($n = 5$ seeds). Positive means corruption hurts prediction. Bracketed intervals are paired bootstrap CIs, shown for scramble and boundary shift, the order- and alignment-sensitive corruptions that are the critical tests for position-aligned use. Both are damaging on the easy source but indistinguishable from zero on harder long-range positions; only wholesale dropout still bites.

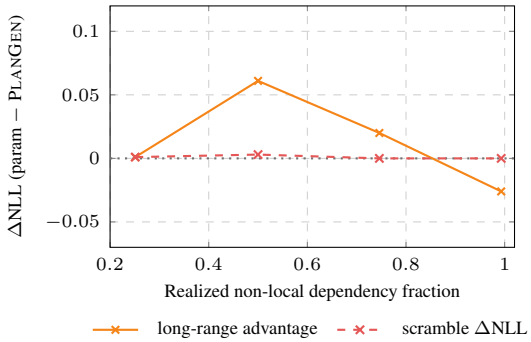

Figure 3: Sweeping the long-range dependency rate on the harder source. For the advantage curve, ΔNLL is param-matched minus PLANGEN, so positive values favor PLANGEN; the scramble curve is the long-range corruption ΔNLL, where positive means the corruption hurts. The advantage is non-monotonic and falls below zero at the most long-range setting, while scramble stays near zero throughout: increasing long-range dependence does not increase position-aligned plan use.

state belongs at which position should be sensitive to order and alignment; the harder-source profile is instead consistent with the plan channel supplying position-agnostic information that helps on average but does not track long-range structure.

Varying the long-range rate gives no causal trend. If long-range dependency were the axis driving plan use, then directly increasing the fraction of genuinely long-range dependencies should strengthen the plan channel's long-range advantage. We test this by sweeping the rate at which governing values are restated inside the local window, holding all other source statistics fixed, and re-running the full confirmation at each level with identical model and training settings. Figure 3 plots the long-range advantage over the equal-parameter control, as param-matched minus PLANGEN NLL so that positive favors PLANGEN, against the realized fraction of non-locally-resolved dependencies. The curve is not monotonic: the advantage is confirmed only at intermediate levels and falls to a zero-crossing interval at the most long-range setting, while the order- and alignment-corruption deltas stay near zero throughout. Sweeping the long-range axis directly does not produce the trend a position-aligned reading would require.

Summary. The same architecture receives different verdicts under the same framework: strong signals of position-aligned use on the easy source, but only an average-likelihood gain on the harder source, where removing plan information still hurts yet corrupting its order or alignment does not. Aggregate likelihood is the metric that fails to reveal this difference. Appendix C.4 summarizes the controls addressed by each result.

6 Conclusion

We studied whether a learned compressed memory is causally used for long-range language modeling, using PLANGEN, a decoder-only model whose explicit block-level plan bank can be directly corrupted. Because the memory is explicit, we could move beyond aggregate likelihood and test causal use through paired interventions on long-range positions. Applied across controlled sources of increasing difficulty, the framework produced opposite verdicts for the same architecture: on an easy templated source the plan channel showed every hallmark of strong causal use, including a 76.4% capacity-diagnostic NLL reduction and uniformly damaging corruptions, while on a harder source with genuine beyond-window dependencies it improved only average likelihood, with no confirmed long-range advantage and order or alignment corruptions that no longer bit, even though removing plan information wholesale still hurt. The large easy-source effects did not carry over to the harder long-range test. We conclude that likelihood gains alone do not establish that compressed memory is used in a position-aligned way for the long-range dependencies it is meant to track, and that explicit intervention tests of the kind we propose are needed to distinguish such use from average-fit improvements. These tests are model-agnostic, and applying them to other compressed-memory architectures is a natural next step.

Limitations

Our evidence comes from two controlled synthetic sources. Controlled sources are what make our central comparison possible, since they let us know which positions are genuinely long-range and corrupt an explicit memory cleanly, but they are simpler than unrestricted natural discourse, and we do not claim our specific numbers transfer to arbitrary corpora. We study one easy and one harder source rather than a continuum of natural text; whether the same dissociation between average and long-range gains appears across many real domains is open.

Our negative long-range finding is a failure to confirm an effect, not a proof that none exists. A larger model, a different memory architecture, or a longer training budget could in principle show confirmed long-range use where ours does not, and our harder-source conclusion rests on the long-range positions and statistical criteria used in this analysis. We mitigate this by holding model, data, and protocol fixed across the difficulty sweep, so the contrast within our study is internally controlled, but it remains a statement about this setting rather than about compressed memory in general.

Finally, our tests target whether memory is causally used for long-range prediction under held-out likelihood and paired corruption. They do not measure open-ended generation quality, downstream task accuracy, or efficiency, and a memory that fails our long-range tests may still be valuable for those other reasons.

Ethics Statement

This work studies whether a learned compressed memory is causally used for long-range language modeling, using small decoder-only models trained on controlled synthetic sources. We see limited direct ethical risk in this setup, but note the following.

Our primary data are synthetic narrative sources generated by a procedure we describe and release. They contain no personal, private, or human-subject data, and no human annotation was collected. The entity names, attributes, and values are drawn from abstract symbol pools and do not refer to real individuals.

Our central result is a failure to confirm an effect: on a harder source, likelihood gains do not translate into confirmed long-range causal use. We have tried to report this honestly, including the settings where the apparent effect was strong, the criteria against which we judged it, and the analyses that did not support our initial hypothesis. We caution against over-reading either the large easy-source gains or the negative long-range result beyond the controlled settings in which they were obtained.

More broadly, our argument is methodological: aggregate likelihood improvements alone should not be taken as evidence that a learned memory is causally used. We hope this encourages more careful evaluation of memory-augmented models, which can reduce overstated capability claims rather than create new risks.

The experiments were run on a single-node GPU setup, and the compute required is modest relative to large-scale pretraining. We report training budgets, seeds, and selection protocols to support reproducibility, and we release the source generator, the generated documents and their gold metadata, and the evaluation code.

References

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14388–14411.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024a. Longalign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1376–1395.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024b. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2025. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: extending llm context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, pages 11091–11104.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations (ICLR)*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *Advances in neural information processing systems*, 35:33248–33261.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Yukyung Lee, Soonwon Ka, Bokyung Son, Pilsung Kang, and Jaewook Kang. 2025. Navigating the path of writing: Outline-guided text generation with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 233–250.

Jiaming Li, Yukun Chen, Ziqiang Liu, Minghuan Tan, Lei Zhang, Yunshui Li, Run Luo, Longze Chen, Jing Luo, Ahmadreza Argha, and 1 others. 2025. Storyteller: An enhanced plot-planning framework for coherent and cohesive story generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20818–20846.

Mo Li, Songyang Zhang, Yunxin Liu, and Kai Chen. 2024. Needlebench: Can llms do retrieval and reasoning in 1 million context window? *arXiv preprint arXiv:2407.11963*.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2024a. Ringattention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*.

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

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. 2025. Nolima: Long-context evaluation beyond literal matching. *arXiv preprint arXiv:2502.05167*.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *International Conference on Learning Representations*.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, and Orhan Firat. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024. Assisting in writing wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, and 5 others. 2024. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*.

Qianyue Wang, Jinwu Hu, Zhengping Li, Yufeng Wang, Daiyuan Li, Yu Hu, and Mingkui Tan. 2024. Generating long-form story using dynamic hierarchical outlining with memory-enhancement. *arXiv preprint arXiv:2412.13575*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. In *International Conference on Learning Representations*.

Haotian Xia, Hao Peng, Yunjia Qi, Bin Xu, Juanzi Li, Lei How, and Xiaozhi Wang. 2025. Storywriter: A multi-agent framework for long story generation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6559–6563.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in neural information processing systems*, 37:119638–119661.

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

Kaige Xie and Mark Riedl. 2024. Creating suspenseful stories: Iterative planning with large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2391–2407.

Xuenan Xu, Jiahao Mei, Chenliang Li, Yuning Wu, Ming Yan, Shaopeng Lai, Ji Zhang, and Mengyue Wu. 2025. Mm-storyagent: Immersive narrated storybook video generation with a multi-agent paradigm across text, image and audio. *arXiv preprint arXiv:2503.05242*.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2024. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. 2024. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*.

	Easy diagnostic	Easy prev-block	Harder long-range
Plan dim d_p	256	512	256
Block size r	32	16	16
Local window k	—	512	512
Plan access	blk-hidden	prev-block	prev-block
Seq. length	2048	2048	2048
Learning rate	2e−4	2e−4	8e−5
Weight decay	0.1	0.1	0.03
Warmup steps	4000	1000	200
Max steps	4000	4000	480
Batch size	1	1	256
Dropout	—	—	0.08
Train docs	44000	50000	6400
Val / test docs	5500	2048	800
Seeds	1	8	5

Table 4: Settings for the reported runs. Plan dimension and block size are selected on validation data per setting. Easy-source runs use the narrative-HMM templated source; the harder long-range run uses the naturalistic source at the most long-range setting (realized non-local fraction 0.99). All runs use hidden size 1024, 24 layers, 16 heads, a byte tokenizer (vocab 256), AdamW, and bf16 mixed precision on H100 80GB GPUs.

A Implementation Details

All models are decoder-only Transformers with hidden size 1024, 24 layers, 16 attention heads, and a byte-level tokenizer with vocabulary size 256, trained with AdamW and gradient clipping 1.0. Plan vectors are produced by a plan encoder (LayerNorm, Linear, GELU, Linear) and fused into the decoder by gating. The equal-parameter local control adds a residual bottleneck adapter sized to match PLANGEN's parameter count, with no block-level plan bank. For each setting, the plan dimension and block size are chosen on validation data, which is why the easy diagnostic and the previous-block confirmation use different values. Table 4 lists the settings behind the reported results.

Parameter counts. On the easy source, the local-window baseline has 304.9M parameters and PLANGEN has 312.8M. The previous-block confirmation compares PLANGEN against an equal-parameter local control matched to the same count, so its +0.003 NLL advantage is not explained by parameter count. The refresh sweep (Section 5.1) holds all settings in the harder-long-range column fixed across dependency-rate levels, varying only the source's non-local dependency fraction.

B Additional Method Details

B.1 Local and Parameter-Matched Baselines

The primary baseline is a local-window decoder-only language model. For position t, the attention mask permits attention to positions j satisfying

$$j \le t \quad \text{and} \quad t - j \le k.$$

We also implement parameter-matched local variants that add non-plan parameters without exposing a block-level plan channel. These variants control for improvements due only to additional parameter count. The reported capacity-control comparison uses the same held-out test documents, tokenizer, split, training budget, and validation-only selection protocol as the previous-block plan comparison.

B.2 Alternative Fusion and Plan Channels

The main experiments use gated fusion:

$$\tilde{h}_t = h_t + g_t \odot W_p r_t, \qquad r_t = \mathrm{Read}(\hat{P}_{\le b(t)}, b(t)). \tag{10}$$

We also implement cross-attention fusion, where token states attend to plan vectors as memory. In addition, we support Gaussian noise and discrete approximations to the plan channel:

$$\tilde{p}_i = \hat{p}_i + \sigma \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, I). \tag{11}$$

Uniform straight-through quantization, vector quantization, and product quantization are implemented for diagnostic experiments, but the main reported results use the strongest continuous or causally restricted configurations.

B.3 Plan-Source Modes

We implement three plan-source modes. The non-causal block-hidden mode computes a plan for block i from hidden states in the same block:

$$\hat{p}_i = f_\phi(\bar{h}_i). \tag{12}$$

This is a plan-capacity diagnostic under teacher forcing. The previous-block mode computes the plan for block i from block $i-1$, preventing current-block or future-block access. The prefix-online mode constructs a prefix summary:

$$s_t = \mathrm{Pool}(h_1, \ldots, h_t), \qquad \hat{p}_t = f_\phi(s_t). \tag{13}$$

For causally restricted experiments, we test that modifying future tokens does not change plans available to earlier positions.

B.4 Intervention Definitions

We evaluate four plan-bank interventions. Swap replaces a document's plan segment with another document's plan segment:

$$\hat{P}'_a[S] = \hat{P}_b[S], \tag{14}$$

where S is a selected block range. Scramble permutes plan order:

$$\hat{P}'_i = \hat{P}_{\pi(i)}. \tag{15}$$

Dropout removes plan vectors with Bernoulli masks:

$$\hat{p}'_i = m_i \hat{p}_i, \qquad m_i \sim \text{Bernoulli}(1-\rho). \tag{16}$$

Boundary shift misaligns token blocks and plan vectors:

$$\hat{p}'_{b(t)} = \hat{p}_{b(t)+\delta}, \tag{17}$$

with clipping at sequence boundaries. All interventions are evaluated in paired form so that the only changed object is the plan bank supplied to the decoder.

B.5 Controlled Continuation Ranking

For each prompt, we construct a gold continuation x^{gold} and a set of corrupted continuations $\mathcal{C}$. Each model assigns teacher-forced NLL to all candidates. We report Gold@1:

$$\text{Gold@1} = \mathbb{I}\left[\text{NLL}(x^{\text{gold}}) < \min_{c \in \mathcal{C}} \text{NLL}(x^c)\right]. \tag{18}$$

We also compute the NLL margin:

$$\text{margin} = \text{NLL}(x^{\text{corrupt}}) - \text{NLL}(x^{\text{gold}}). \tag{19}$$

Positive margins indicate that the model assigns lower NLL to the state-consistent continuation.

C Controlled Source Details

C.1 Narrative-HMM Generator

The primary controlled source is a narrative-HMM generator. Each document is generated from a global genre variable G, an entity table E, and a sequence of block-level latent states z_i:

$$G \sim p(G), \tag{20}$$

$$E \sim p(E \mid G), \tag{21}$$

$$z_0 \sim p(z_0 \mid G), \tag{22}$$

$$z_i \sim p(z_i \mid z_{i-1}, G) \quad \text{for } i > 0. \tag{23}$$

Given (G, E, z_i), the generator renders an event tuple u_i and lexicalizes it into a text block x_i. A document is the concatenation

$$x_{1:T} = x_0 \oplus x_1 \oplus \cdots \oplus x_{B-1}. \tag{24}$$

Evidence	Key control	Alternative addressed
Block-hidden diagnostic	local-window and plan grid	gain not from local block size alone
Equal-parameter control	equal-parameter local adapter	gain not explained by parameter count
Plan-bank corruptions	paired plan corruption	plan path is not ignored
Corruption severity	severity sweep	degradation is graded, not incidental
Previous-block test	no current/future block access	gain not from current-block leakage
Continuation ranking	gold vs. corrupted candidates	tracks state-consistent continuations
Harder-source long-range	long-range-position NLL and corruptions	separates average gain from long-range use

Table 5: Evaluation controls and alternative explanations addressed by the results.

C.2 Entity and Event Metadata

Each document contains an entity table with roles, attributes, goals, or locations. The generator records gold entity tuples

$$(e, a, v), \tag{25}$$

where e is an entity, a is an attribute, and v is the current value. It also records coreference links and contradiction spans. These annotations are not used as language-model supervision, but support diagnostic analysis and controlled corruption construction.

C.3 Surface Variation

The lexicalizer injects paraphrase noise, genre-specific filler, and controlled contradiction templates. Paraphrase noise changes surface realization while preserving the underlying entity or event state. Filler increases local fluency demands without changing the core latent state. Contradiction templates intentionally violate earlier attributes or commitments, enabling controlled continuation-ranking and corruption tests.

C.4 Evaluation Controls

Table 5 summarizes the main controls used to address alternative explanations for the results.

Model	Plan dim.	Block size	NLL
Local baseline	–	128	0.139646
Local baseline	–	32	0.140402
PLANGEN block-hidden	64	128	0.098981
PLANGEN block-hidden	64	32	0.052113
PLANGEN block-hidden	128	128	0.083719
PLANGEN block-hidden	128	32	0.079629
PLANGEN block-hidden	256	128	0.119049
PLANGEN block-hidden	256	32	0.032944

Table 6: Controlled synthetic held-out grid. Figure 2 highlights the best local and best non-causal block-hidden PLANGEN models; this table shows the full grid over plan dimension and block size.

C.5 Gaussian Block Source

We also implement a Gaussian block source as a diagnostic stress test:

$$z_0 \sim \mathcal{N}(0, I), \tag{26}$$

$$z_{i+1} = Az_i + \epsilon_i, \tag{27}$$

$$y_i = Wz_i + \eta_i. \tag{28}$$

This source is used for information-channel diagnostics and scaling-law code paths, not for the main benchmark.

C.6 Splits and Leakage Checks

Synthetic documents are split into training, validation, and held-out test sets. Model selection is performed on validation metrics, and final comparisons are reported on held-out documents. For causally restricted experiments, we verify that future tokens do not change earlier available plans. Intervention and continuation-ranking evaluations use paired examples so that clean and corrupted variants differ only in the supplied plan bank.

C.7 Controlled Synthetic Grid

Table 6 reports the full grid over plan dimension and block size for the non-causal block-hidden diagnostic. The best local baseline is block size 128, while the best block-hidden PLANGEN model uses plan dimension 256 and block size 32.