Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Similarity Finds the Fact, Not the Version: A Co-Trained Order Stamp in the Key Resolves Version Conflict in Persistent Model-Internal Memory

Maximiliano Rodrigo Speranza

maximiliano.speranza@gmail.com

Independent Researcher https://orcid.org/0009-0005-0413-8554

Research Article

Keywords: language models, persistent memory, retrieval, version conflict, temporal metadata, proactive interference, pre-registration, small language models

Posted Date: September 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10896018/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Similarity Finds the Fact, Not the Version

A Co-Trained Order Stamp in the Key Resolves Version Conflict in Persistent Model-Internal Memory

Maximiliano Speranza
Independent Researcher, Buenos Aires, Argentina
ORCID 0009-0005-0413-8554
maximiliano.speranza@gmail.com

September 2026

Abstract

A persistent archive co-trained inside a language model learns to retrieve the right fact and then fails to know which *version* of it currently holds. We measure that failure exactly: with one version archived the model answers 0.9974, and with two versions of the same key competing it drops to 0.4576, which is chance between the old value and the new one. The retrieval is intact and the ordering is absent.

We show that the missing information fits in a small co-trained **order stamp added to the archive key**, and that it is the ordering, not the extra capacity, that does the work. Across five seeds, revised keys go from 0.4570 ± 0.0289 without a stamp to $\mathbf{0.9956 \pm 0.0029}$ with one. The falsification control, an identically parameterised stamp whose value bears no relation to the real write turn, gives 0.4768 ± 0.0389, within one standard deviation of the baseline. The fifteen runs order perfectly by condition with no overlap between cells.

Three further results sharpen the claim. First, with three versions per key, each written in its own sequence so that no recency trace survives in the content, asking for the *superseded* value goes from 0.2917, which is chance among three, to 0.8316. Second, when write turns are sampled at random so that no fixed slot-to-role table can solve the task, the stamp still yields 0.9644 on the current version, which rules out the table shortcut. Third, and practically the most useful, a *corrupted* stamp is worse than no stamp at all, degrading even item identification from 0.9974 to 0.8802. A wrong timestamp is not missing information, it is damaging information.

We also separate two capabilities that are usually conflated. Preferring the latest value and using the order are learned at different times, the first saturating around 3,000 steps and the second only emerging near 4,000 and saturating near 6,000. A benchmark that only asks for the current value cannot see the distinction, because the recency shortcut already answers it.

1 The problem, measured

Linear-attention and delta-rule memories have a fixed-size recurrent state and overwrite on write, so what is old is lost by construction (Schlag et al., 2021; Yang et al., 2025). The natural repair is an external archive, and a growing line of work co-trains that archive inside the model rather than bolting a frozen retriever onto it (Feldman et al., 2026; Jeong, 2026; Zhao et al., 2025).

That repair works for retrieval and leaves a specific hole. In our own prior stage, a co-trained archive holding vectors produced by the model itself reaches 0.9974 when a key has a single archived version, and collapses to 0.4576 when the same key has two versions competing. The value 0.4576 is not noise, it is chance between the old and the new. The model finds the right fact and does not know which version rules.

This is the same failure mode we had already measured in a non-parametric index, where geometry grouped versions of a memory perfectly and recovered the *oldest* one. Two mechanisms

with nothing in common fail in the same way, which suggests the diagnosis is a property of similarity-based recall and not of any particular implementation. Similarity identifies the item. Order is information similarity does not contain.

2 Related work

Intra-sequence hybrids. Cui (2026) pairs a linear-attention state with an exact memory for what that state forgets, and Lufkin et al. (2026) combines a recurrent state with attention that supplements it only where the state predicts poorly. Both are hybrids of a compressive state with an exact store, and in both the store lives inside one forward pass, so the question of which of two versions written in *different* sessions holds cannot arise.

Co-trained persistent memory. Co-LMLM (Feldman et al., 2026) is the closest in mechanism, emitting continuous queries from the model's hidden states to retrieve from a knowledge base, at 135M to 360M parameters. Trained Persistent Memory (Jeong, 2026) accumulates a memory bank across turns on a frozen GPT-2 backbone so facts stated in one session are recalled later. Neither versions a fact, neither stamps order into the key, and neither ever asks for a superseded value. Co-LMLM treats the base as a static snapshot; Jeong (2026) applies a uniform temporal decay factor, which attenuates old entries rather than ordering them.

Proactive interference. Wang and Sun (2025) is the closest in *task*, updating keys several times within a context and asking for the current value with old versions present as interference. It documents the same recency shortcut we find. It is a benchmark and a diagnosis, its authors state that they offer no architectural mitigation, it evaluates only large pretrained models, and it never asks for the previous value. That last point matters for us, because the recency shortcut answers the current-value question by itself.

Application-layer versioning. Systems that version documents with explicit metadata and pointers do exist, in retrieval pipelines built on top of a frozen model. The stamp we study is not that: it lives inside the model's own retrieval key and is trained jointly with the reader, rather than sitting in a metadata field consumed by a separate retriever.

Timestamps in the input. Rajaby Faghihi and Kordjamshidi (2021) injects time markers into the input for event streams, past, present and future. That is not two versions of the same fact inside a memory, and the marker does not enter a retrieval key.

3 Setup

The substrate is a small language model with a co-trained archive, 64 hidden dimensions, an archive of 9 entries, and a synthetic cross-sequence task. Facts are written in one sequence, revised in another, and queried in a third, **with the recurrent state reset between sequences**. Retrieval therefore cannot be solved by carrying activations forward, and the floor is chance by construction.

The archive stores vectors produced by the model, not tokens. Writing entry i from state h_i gives key $k_i = h_i W_k$ and value $v_i = h_i W_v$; reading forms a query from the consuming state and attends over the archive. The **stamp** adds a learned embedding of the write turn to the key,

$$k_i \;=\; h_i W_k \;+\; \mathrm{ord}[\,t_i\,],$$

where t_i is the turn at which entry i was written and ord is a trained table. It is 384 effective parameters. Crucially, **the archive is shuffled per sample** with each stamp travelling attached to its own entry. Without shuffling, write turn would coincide with tensor position and the experiment would measure "look at the end of the archive", which is position and not order.

Three conditions are compared throughout. `none` has no stamp. `stamp` is as above. `shuffled` is the falsification control, with an identically sized and identically trained table whose stamp value bears *no relation* to the real write turn.

4 Result 1: the stamp lifts version conflict, and it is the ordering that lifts it

Five seeds, two versions per revised key, 4,000 steps.

condition	global	**revised keys**	unrevised keys
`none`	0.7273 ± 0.0146	0.4570 ± 0.0289	0.9977
`stamp`	$\mathbf{0.9970 \pm 0.0020}$	$\mathbf{0.9956 \pm 0.0029}$	0.9984
`shuffled`	0.7374 ± 0.0194	0.4768 ± 0.0389	0.9979

The gain over no stamp is +0.5385 and over the shuffled control +0.5187. **The five stamped seeds fall between** 0.9909 **and** 0.9987**, and the best seed of the other two conditions reaches** 0.5156**. The fifteen runs order perfectly by condition with no overlap.** At that separation a confidence interval is a formality.

The cell that did not win is what makes the result valid. `shuffled` has exactly the same parameters as `stamp`, trained identically, and differs only in that the stamp carries no relation to the real write turn. Without it, +0.5385 would be explained just as well by saying we gave the reader more capacity. With it, the capacity explanation is ruled out directly.

The arithmetic is symmetric. Version conflict cost −0.5398 and the stamp returns +0.5385. It restores what similarity lost and nothing more, leaving unrevised keys at 0.998 where they already were.

5 Result 2: three versions, and asking for the superseded value

A first attempt at three versions broke itself, and the way it broke is worth recording. Writing both revisions inside the same sequence let the state at v_3 carry a trace of v_2, so order was written into the *content* without any metadata putting it there. At 4,000 steps this was invisible because no condition solved the hard question; at 12,000 steps the no-stamp condition reached 0.97 and exposed the leak.

With each version written in its own sequence and the state reset between them, v_2 and v_3 are each "first and only mention of their sequence", indistinguishable in recency. The only thing in the system that says which came first is the stamp. Three seeds, 12,000 steps.

condition	current	**previous**	single-version keys
`none`	0.3090	0.2917 ± 0.0207	0.9436
`stamp`	0.9774	$\mathbf{0.8316 \pm 0.2399}$	0.9974
`shuffled`	0.2682	0.2804 ± 0.0159	0.8802

Without a stamp the model sits at 1/3, chance among three versions, while still identifying the item when there is no conflict (0.9436). The leak is closed, and the distance from the 0.97 of the broken design measures exactly how much order had been leaking through content.

This number is budget-dependent and we report it as such. With five seeds the previous-value score falls to 0.7667 (sd 0.2340): 0.9766, 0.9635, 0.5547, 0.8594, 0.4792. The mechanism does not fail, two of five seeds have not converged at 12,000 steps. Tested directly, seed 2 extended to 24,000 steps moved from 0.5547 to 0.9531. The honest verdict is conditional: the stamp allows answering about the previous version, at a budget that 12,000 steps does not always reach. **In this regime convergence time is part of the phenomenon, not an implementation detail.**

6 Result 3: it is not a slot-to-role table

In the previous design the write turns were fixed per role, so the model could learn "embeddings $9, 10, 11$ mean current". That is a slot-to-role correspondence, not an order, and the shuffled control does not separate them because randomising the stamp breaks the table just as it breaks the order.

We therefore sampled twelve turns at random out of a range of thirty-two per sample, sorted them, and assigned them respecting the real write order. Turn 9 can be a first version in one sample and a third in the next, so no fixed table solves the task. Three seeds.

condition	**current**	previous
`none`	0.3247	0.2977 ± 0.0326
`stamp`	**0.9644**	0.3142 ± 0.5330
`shuffled`	0.2804	0.2925 ± 0.0266

The current-version result survives the moving turns, which **rules out the table shortcut and validates Result 1 backwards**: had the mechanism been a table, moving the turns would have broken the current version too.

The previous-value column is **bimodal, not noisy**: 0.0052, 0.9297, 0.0078. Two seeds fall into a recency shortcut, answering *below* chance because they return the current value when asked for the previous one, and one seed learns the full comparison. We report the distribution rather than the mean, because the mean of 0.31 describes none of the three runs. The honest summary is that the reader learns to read the clock to know what time it is now, not to reconstruct the sequence of what happened.

7 Result 4: a lying stamp is worse than no stamp

Reading the `shuffled` row of Result 2 as a result in itself, rather than only as a control, gives the finding with the most practical bite. A corrupted stamp leaves the current version at 0.2682, *below* the 0.3090 of having no stamp at all, and it degrades item identification from 0.9974 to 0.8802, a capability that has nothing to do with ordering.

The reader leans on the stamp. Once it does, a corrupted timestamp is not missing information, it is damaging information, and the damage spreads to retrieval quality. For any system that stamps memory entries with a write time, this says that a stale or wrong stamp is worse than an absent field.

8 Two capabilities, learned at different times

Preferring the latest value and using the order are distinct and separable, and the ordering of the two is stable across replicates even though the timing is not. In both replicates at $lr = 10^{-3}$ the current version saturates by 3,000 steps (0.9583 in both) while the previous-value score is still 0.0000. The previous-value score then lifts off later, and *how much* later varies: one replicate reaches 0.2500 at 4,000, 0.6250 at 5,000 and 0.9583 at 6,000, while the other is still at 0.0208 at 4,000 and does not pass 0.9 until 8,000. We report both because the mean of the two would describe neither. What replicates is the **ordering**: in every run the current version is solved thousands of steps before the previous one, and at the step where the current version is already at 0.9583 the previous one is at exactly 0.0000.

This has a direct consequence for evaluation. **A benchmark that only asks for the current value cannot see the distinction**, because the recency shortcut gives the right answer. This is precisely what Wang and Sun (2025) does not do, and it is why we consider the previous-value question the more informative probe.

9 Limitations

Sufficiency, not inference. In this task the current version is always the one with the highest turn, so the learnable policy is "take the largest stamp". Our result says the information similarity lacks *fits in a stamp* and that the reader can use it. It does not say the model reasons about time.

The stamp is given, not inferred. We show the reader can use the metadata, not that it can fabricate it. This is not an unrealistic oracle, since write turn is free to any index, unlike knowing which version is correct, but it does bound the claim.

Scale. A 64-dimensional model, a 9-entry archive and a synthetic task. The measured difficulty is alignment and version conflict, not discrimination among many candidates.

A fresh archive. The archive is recomputed at every step in these experiments. A follow-up in which entries age shows the damage is gradual rather than a cliff, with revised keys falling from 0.9987 to 0.9115 as the cosine between write-time and read-time frames falls to 0.78.

10 Conclusion

Similarity identifies the item; order is information that similarity does not contain and that has to be carried separately. In a persistent memory co-trained inside a small language model, carrying it as a stamp in the key takes version conflict from chance to near ceiling, and a control with the same parameters and a meaningless stamp shows the ordering rather than the capacity is doing the work. The same mechanism makes the superseded value answerable, which no benchmark in this area currently asks for, and it comes with a warning that generalises past our setup: once a reader learns to lean on a stamp, a wrong stamp costs more than no stamp.

Reproducibility and data availability

Every experiment reported here was pre-registered with its predictions and thresholds hashed before the corresponding run. Cells that failed their pre-registered criteria are reported as failures, including the budget-dependent previous-value score of Result 2 and the bimodal column of Result 3. All pre-registrations with their SHA hashes, analysis scripts, per-seed raw results and machine-generated reports are archived in the project repository, `https://github.com/SperanzaMax/delta-archivo`, whose commit history carries the timestamps that establish the order of pre-registration and result.

Declarations

Funding. None. **Competing interests.** None. **Ethics approval.** Not applicable; no human or animal subjects. **Data availability.** See above. **Author contributions.** Sole author.

References

Cui, W. (2026). A Hippocampus for Linear Attention: An Exact Memory for What the Recurrent State Forgets. arXiv:2607.02303.

Lufkin, L., Figliolia, T., Millidge, B., and Krishnamurthy, K. (2026). Hybrid Associative Memories. arXiv:2603.22325.

Feldman, Y., Zhao, L., Godey, N., Go, D., Hua, Y., Weinberger, K. Q., Sun, J. J., and Artzi, Y. (2026). Co-LMLM: Continuous-Query Limited Memory Language Models. arXiv:2607.07707.

Jeong, H. (2026). Trained Persistent Memory for Frozen Decoder-Only LLMs. arXiv:2603.22329.

Zhao, L., Zalouk, S., Belardi, C. K., Lovelace, J., Zhou, J. P., Noonan, R. T., Go, D., Weinberger, K. Q., Artzi, Y., and Sun, J. J. (2025). Pre-training Limited Memory Language Models with Internal and External Knowledge. arXiv:2505.15962.

Wang, C., and Sun, J. V. (2025). Unable to Forget: Proactive Interference Reveals Working Memory Limits in LLMs Beyond Context Length. arXiv:2506.08184.

Schlag, I., Irie, K., and Schmidhuber, J. (2021). Linear Transformers Are Secretly Fast Weight Programmers. ICML 2021.

Yang, S., Kautz, J., and Hatamizadeh, A. (2025). Gated Delta Networks: Improving Mamba2 with Delta Rule. ICLR 2025. arXiv:2412.06464.

Rajaby Faghihi, H., and Kordjamshidi, P. (2021). Time-Stamped Language Model: Teaching Language Models to Understand the Flow of Events. NAACL 2021. arXiv:2104.07635.