Runtime Working Memory Editing in Linear Recurrent Models: A Simple Method and Its Limits

Martin Chang
AINekko
martin@nekko.ai

April 24, 2026

Abstract

We investigate whether factual associations stored in the recurrent state of the RWKV language model can be edited at inference time. We show that, given two short calibration prompts that differ only in the target fact, a simple state delta can be added to the live recurrent state to rewrite the model's in-context answer—without gradient updates or weight modification. On RWKV-7 7.2B, this method succeeds on clean single-fact edits, achieving roughly two-thirds accuracy while largely preserving unrelated facts.

However, the effect is brittle. Accuracy drops sharply when the context includes prior overwrites, when multiple properties are edited jointly, when evaluation requires one-hop inference, or when semantically related filler turns are inserted after the edit. To systematically evaluate these behaviors, we introduce RIME, a benchmark for runtime internal memory editing with controlled axes including distractors, overwrites, composition, multi-property edits, stacking, and multi-turn persistence.

Our main conclusion is empirical: while RWKV's recurrent memory supports a simple runtime editing operation, the resulting edits are not robust under modest perturbations.

On the full RIME test split, the edit succeeds in 45% of cases while preserving 92% of unrelated in-passage facts. In the cleanest single-fact setting, accuracy rises to 67%, and to 92% when calibration is template-aligned.

1 Introduction

Neural language models encode factual knowledge in their parameters [Petroni et al., 2019]. Editing this parametric memory typically requires methods such as ROME [Meng et al., 2022] or MEMIT [Meng et al., 2023], which compute weight updates via constrained optimization. Such edits are persistent, affect all downstream uses of the model, and require backpropagation.

Recurrent architectures such as RWKV [Peng et al., 2023, 2024] offer a different locus of memory: a recurrent state that evolves across tokens and accumulates information from the input. For example, when a user states "Alice lives in Paris" in a dialogue, this association is encoded in the recurrent state as part of standard forward computation. This working memory is distinct from parametric memory: it is computed at runtime and can, in principle, be directly manipulated through inference-time state access.

In this work, we explore whether such runtime memory can be edited. Our method modifies in-context factual associations by adding a precomputed state delta to the recurrent state:

$$\begin{aligned} \text{wkv} &\leftarrow \text{wkv} + \left(\text{wkv}_{\text{new}} - \text{wkv}_{\text{old}}\right) \\ x &\leftarrow x + \left(x_{\text{new}} - x_{\text{old}}\right) \end{aligned} \tag{1}$$

where the subscripted states are obtained from short calibration prompts that differ only in the target fact. This operation simultaneously updates the wkv state (the time-mix recurrent

memory) and the residual stream, requires no gradient computation, and takes effect immediately at inference time.

Empirically, we find that this simple mechanism can rewrite single in-context facts in RWKV-7 7.2B with moderate success. However, the effect is fragile: performance degrades under prior overwrites, multi-property edits, compositional queries, and even modest semantic distraction. These limitations suggest that while recurrent state is editable, it does not support robust fact manipulation under realistic conditions.

Contributions.

- We show that a single state-delta vector, derived from two short calibration prompts, can rewrite in-context factual associations in RWKV-7. To our knowledge, this is the first demonstration of such a simple runtime editing mechanism in linear recurrent language models. We emphasize this as an empirical observation rather than a fully reliable method.
- We introduce **RIME** (Runtime Internal Memory Editing), a benchmark that isolates key axes of difficulty—including distractors, overwrites, composition, multi-property edits, stacking, multi-turn filler, and target position—allowing failure modes to be analyzed systematically.
- We characterize the method's performance, identifying regimes where it succeeds (clean single-fact edits) and where it fails (overwrites, joint edits, compositional queries, and semantically related filler). We position these findings as a probe of recurrent working memory and a starting point for future work on robust runtime editing.

2 Background

RWKV [Peng et al., 2023, 2024] is a linear recurrent language model that processes tokens sequentially while maintaining a fixed-size state. We focus on RWKV-7 [Peng et al., 2025], which has two state components per layer. The first is the wkv state, a time-mix recurrent memory. Each layer ℓ has H attention heads, each maintaining a $d \times d$ matrix (d = head size). Following the notation of Peng et al. [2025], the state evolves as:

$$\text{wkv}_t = \text{wkv}_{t-1}\big(\text{diag}(w_t) - \hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)\big) + v_t^T \cdot \hat{k}_t \tag{2}$$

where $w_t \in (0,1)^d$ is a data-dependent decay, $\hat{k}_t$ and $\hat{\kappa}_t$ are the replacement and removal keys, v_t is the value vector, and a_t is the in-context learning rate. Each token writes approximately a rank-1 update; the decay w_t controls how quickly old information fades.

The second component is a token-shift state: a per-layer buffer carrying the previous token's embedding into the current step (Eq. 3 in Peng et al., 2025): $x_t^{\square} = \text{lerp}(x_t, x_{t-1}, \mu_{\square})$. This provides a one-step temporal context to each layer's weight preparation.

For a model with L layers, H heads per layer, and head size d, the resulting state geometry is:

- wkv: $L \times H \times d \times d$ floats (RWKV-7 7.2B: $32 \times 64 \times 64 \times 64 = 8.4$M floats, ≈ 33 MB at fp32)
- Token-shift: $L \times 2 \times D$ floats (D = model dimension) (RWKV-7 7.2B: $32 \times 2 \times 4096 = 262$k floats, ≈ 1 MB at fp32)

3 Motivation

Before attempting to edit the recurrent state, we first ask a basic question: are in-context factual associations actually encoded there in a way that can be causally manipulated? If not, runtime editing would have no clear target.

Table 1: Causal state transplantation on RWKVv7 7.2B. $P(\text{old})$ after transplanting layer ranges from prompt B's state into prompt A's state. Bold indicates the fact flipped ($P(\text{new}) > 0.8$). 'Baseline" is prompt A's unmodified state. 'Ground truth" is prompt B's unmodified state (the target after a perfect edit). The "wkv only" row transplants only the time-mix state without the token-shift buffer.

Layer range	Hat color	City	Animal	Number
Baseline	0.997	0.999	0.979	0.996
Ground truth	**0.004**	**0.002**	**0.188**	**0.007**
ALL	**0.004**	**0.002**	**0.188**	**0.007**
L0–15	0.997	0.999	0.981	0.996
L16–31	**0.004**	**0.002**	**0.182**	**0.008**
L24–31	**0.053**	**0.008**	0.301	**0.062**
L16–31 wkv only	**0.004**	**0.002**	**0.178**	**0.008**

To test this, we perform a causal state transplantation experiment. We construct two prompts, A and B, that differ in a single fact (e.g., "Alice has a red/green hat"), process them independently, and capture their recurrent states. We then transplant selected layers from B's state into A's state and measure whether the model's output shifts from the original fact to the new one.

Table 1 reports $P(\text{old})$, the probability assigned to the original answer token, after transplanting different layer ranges. Transplanting the upper half of layers consistently flips the model's prediction across all tested fact types, while the lower layers have negligible effect. Moreover, transplanting only the wkv component (without the token-shift state) yields nearly identical results in this setting.

These results establish that the recurrent state is a viable target for editing: factual information is present, localized (primarily in upper layers), and causally influences the model's output. This suggests that modifying the state—rather than the model parameters—may be sufficient to alter in-context behavior.

The remaining question is whether such edits can be achieved *without* access to a full alternative prompt, using only a minimal calibration signal.

4 Method

Our approach edits in-context factual associations by directly modifying the RWKV recurrent state at inference time. The core idea is simple: we estimate the state change corresponding to replacing one fact with another, and then apply this change to a live state. Concretely, given a source fact (e.g., 'Alice lives in Paris") and a target fact ('Alice lives in London"), we construct two short calibration prompts that differ only in the target fact. Both prompts are wrapped in the same chat template and share the same initial state.

We run each prompt independently to obtain their resulting recurrent states, including both the wkv component and the residual stream (token-shift buffer). The difference between these states isolates the effect of the factual change:

$$\Delta\text{wkv} = \text{wkv}_{\text{new}} - \text{wkv}_{\text{old}} \qquad \Delta x = x_{\text{new}} - x_{\text{old}} \tag{3}$$

We then apply this delta to a target state at inference time:

$$\text{wkv} \leftarrow \text{wkv} + \Delta\text{wkv} \qquad x \leftarrow x + \Delta x \tag{4}$$

This operation modifies the model's in-context behavior immediately, without gradient updates or weight changes.

Algorithm 1 Runtime working memory edit

Require: Target state (wkv, x)
Require: Edit prompts p_{old}, p_{new}
Require: Chat template T
$p'_{\text{old}} \leftarrow T(p_{\text{old}})$; $p'_{\text{new}} \leftarrow T(p_{\text{new}})$ {wrap in chat template}
$(\text{wkv}_{\text{old}}, x_{\text{old}}) \leftarrow \text{forward}(p'_{\text{old}})$
$(\text{wkv}_{\text{new}}, x_{\text{new}}) \leftarrow \text{forward}(p'_{\text{new}})$
$\Delta\text{wkv} \leftarrow \text{wkv}_{\text{new}} - \text{wkv}_{\text{old}}$; $\Delta x \leftarrow x_{\text{new}} - x_{\text{old}}$
for layer ℓ in all layers **do**
  $\text{wkv}^{(\ell)} \mathrel{+}= \Delta\text{wkv}^{(\ell)}$
  $x^{(\ell)} \mathrel{+}= \Delta x^{(\ell)}$
end for

Because both calibration prompts share the same initial state and surrounding context, their difference cancels shared components and isolates the directional change associated with the fact. The resulting delta acts as a local "edit vector" in state space. Including relevant context (e.g., nearby entities or narrative details) in both prompts improves edit quality by ensuring better alignment between the calibration and target states.

Motivated by the causal tracing results in Section 3, we apply the edit across all layers $\ell \in 0, \ldots, L-1$. In practice, the effect is driven primarily by upper layers, but applying the delta uniformly is simple and effective.

Finally, the representation of a fact in the recurrent state depends on its token context. In particular, processing a fact as raw text yields a state that differs from processing the same fact within a chat template. Since the live state being edited is produced under a specific template, the calibration prompts must use the same format for the resulting deltas to align. We quantify this effect in Section 5.6.

5 Evaluation

5.1 Overview

We evaluate whether runtime state edits successfully rewrite in-context facts while preserving unrelated information. Our evaluation is designed to answer two questions: (i) when does the edit succeed, and (ii) how robust is the edited state under realistic perturbations?

To this end, we measure generation accuracy after editing and introduce a benchmark, **RIME**, that systematically varies factors known to challenge in-context memory.

5.2 Setup

We evaluate on RWKV-7 7.2B (G1E, 32 layers, 64 heads). We focus on a single model size to isolate the editing mechanism and its failure modes rather than perform a scaling study.

Our primary metric is generation accuracy: after applying the edit, we prompt the model and check whether the generated answer reflects the new fact. For inference-style queries (§5.3), we accept any element of a curated alias set (e.g., 'child", 'pediatric", "kid").

We avoid token-level probability metrics such as $P(\text{old}) = P(t_{\text{old}})/(P(t_{\text{old}}) + P(t_{\text{new}}))$, which we find unreliable: models can assign low probability to the old token while still generating it in free-form output. This issue is exacerbated for multi-token values, where only the first token is measured.

RWKV checkpoints additionally emit intermediate reasoning traces (e.g., `<think>` blocks) inline with generation. These traces often contain the edited value even when the final answer does not. To avoid overestimating accuracy, we report results on `generation_stripped`, where

such spans are removed. Both raw and stripped correctness are logged per example; all reported results use the stripped metric.

5.3 The RIME Benchmark

To evaluate robustness, we introduce **RIME** (Runtime Internal Memory Editing), a benchmark that decomposes each test case into two components. Each case consists of: (i) a *recipient* passage that establishes the original fact and builds the live state, and (ii) an *edit* pair (source/destination) that defines the desired change.

Crucially, these are written independently. The edit pair uses a different scene and vocabulary from the recipient, so the resulting state delta must generalize across contexts rather than exploit surface overlap. Earlier versions that reused the recipient text yielded artificially high accuracy; this decoupling removes that shortcut.

Each case is also annotated with a vector of control variables that isolate specific failure modes:

- **distractors**: competing entities with different values
- **overwrites**: prior values replaced over time
- **composition**: direct vs. one-hop inference queries
- **multi-property edits**: editing several attributes jointly
- **stacking**: sequential application of multiple edits
- **multi-turn persistence**: filler dialogue between edit and query
- **context richness and edit style**: variation in linguistic form
- **target position**: order of mention within the passage

This design allows performance to be analyzed along individual axes rather than collapsed into a single aggregate score.

Recipients are generated to contain the original fact (and any requested distractors or overwrites). Edit pairs share only the target binding and vary in linguistic richness. Each case is retained only if the unedited model correctly answers the query, ensuring that failures reflect the editing mechanism rather than passage ambiguity.

We additionally include *probe queries* about unrelated facts in the same passage to measure the edit's *blast radius*.

The seven control axes define a joint grid of 864 cells, reduced by roughly a third once domain-motivated exclusions are applied (composition is incompatible with overwrites, multi-property edits, and stacking; conflict markers are incompatible with overwrites and composition). Filling this grid uniformly is infeasible at our scale: the test split contains 215 cases, so average occupancy is well below one per cell, and higher-order interactions cannot be estimated directly.

We therefore sample to balance axis *marginals* rather than joints. Each case is drawn greedily from the admissible pool, picking at every step the cell that minimises the sum of already-used per-axis counts, with random tie-breaking. The dev split additionally stratifies by (difficulty $\times$ has-distractors) and, within each stratum, balances the editing target's position so that position effects are not confounded with the harder axes.

This design supports the one-way axis cuts reported in §5.5 but leaves residual joint imbalance that can colour individual effects. The composition/direct gap is the clearest example: exclusion rules confine composition cases to the easier region of the grid, so part of the observed gap is a selection artefact rather than a property of the edit. Two-axis claims are restricted to the coverage shown in Fig. 4.

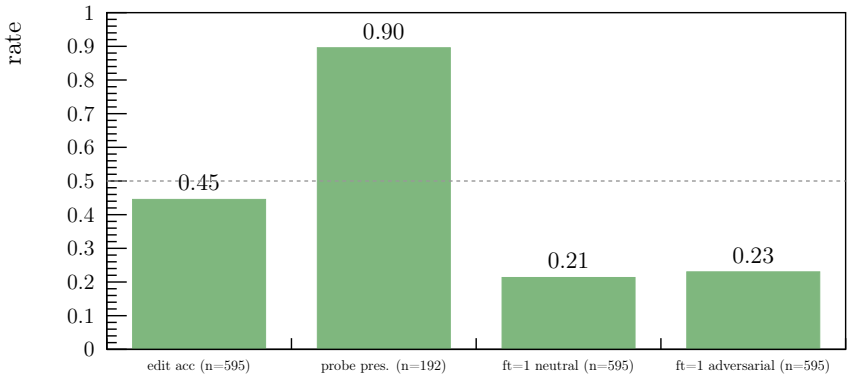

Figure 1: **Headline summary (pooled).** From left to right: edit accuracy at ft=0; probe preservation on unrelated facts in the same passage; edit accuracy after one neutral filler turn; edit accuracy after one adversarial filler turn. Dashed line at 0.5. The edit lands on a minority of cases, but when it lands it is local; a single intervening turn roughly halves its effect.

5.4 Evaluation protocol

For each case, we: (1) construct the recipient state, (2) apply the edit, (3) optionally insert k filler turns, and (4) query the model.

We report accuracy on the edited fact and on probe queries, and analyze results as a function of the RIME control axes.

5.5 Results

We organize the results around four questions: (i) how well does the edit work on average, (ii) which axes drive the difficulty, (iii) does the edit persist across turns, and (iv) when the edit lands, is it local. We present one figure per question (Figs. 1–4) on the RIME test split (595 edit rows at ft=0, 503 probe rows); all accuracies use `correct_stripped` scoring (§5.2).

Figure 1 summarizes the method in four numbers. Pooling across every RIME condition, the state-delta edit rewrites the in-context answer 45% of the time. Under the same conditions, probe queries on unrelated facts in the same passage are answered correctly 92% of the time, indicating that when the edit lands it is largely confined to the target binding. Persistence is weak: inserting a single filler turn between the edit and the query—whether the filler is topically neutral or semantically adjacent to the edited attribute—drops accuracy to roughly 22%. The editable signal is real, but it decays within one recurrent update of unrelated content.

In the cleanest setting — single-property edits with no overwrites or distractors, which is a small corner of the overall RIME benchmark dataset across all difficuly axises — edit accuracy reaches 67% (14/21) on the test split. This aggregate hides a strong dependence on calibration style: when the calibration prompt matches the recipient's chat-template context, 11 of 12 cases succeed (≈92%), paired with 92% probe preservation; a short, out-of-context calibration sentence accounts for nearly all of the remaining failures (3/9). In the absence of interference and with an aligned calibration prompt, the method is therefore near-ceiling.

Beyond this best case, the dominant driver of difficulty is the amount of competing in-context material. Figure 2 breaks edit accuracy down by RIME control axis. Distractor count shows a clean monotonic drop (55% at zero distractors, 48% at two, 32% at four), making it the strongest single-axis effect. Calibration style also matters: template-aligned short and narrative forms perform similarly (48–50%), while the short out-of-context form drops to 36%, consistent with the alignment argument of §4. The remaining axes are comparatively weak—overwrite count is non-

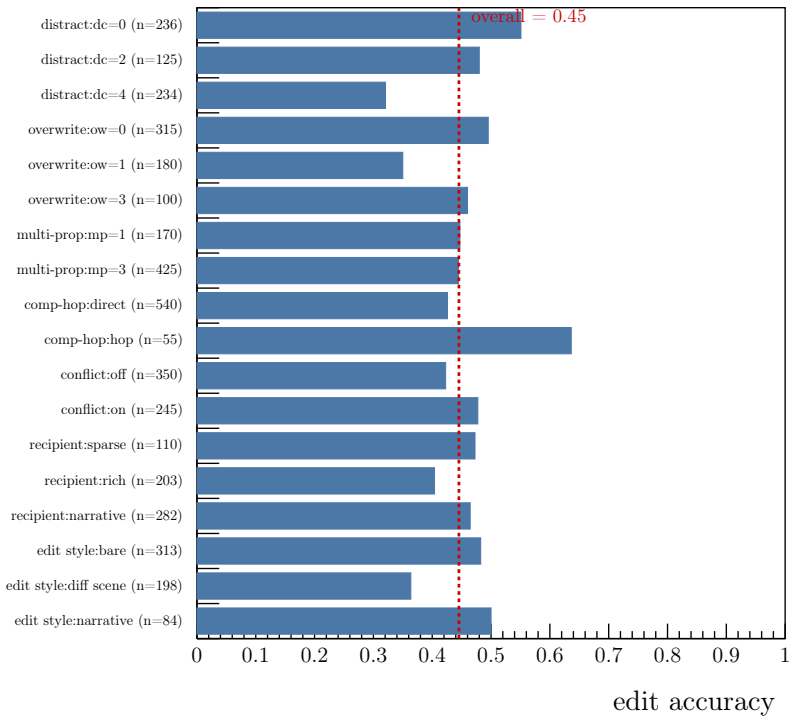

Figure 2: **Difficulty forest (ft=0).** Edit accuracy for every axis value of every RIME control axis. Vertical red line is the pooled overall accuracy (0.45). Distractor count and calibration style carry the largest single-axis effect; overwrite count, multi-property count, conflict marker, and recipient richness are flatter than the design anticipated. The hop vs. direct gap is dominated by selection on the other axes (see text).

monotonic (50/35/46%), multi-property count is essentially flat (∼45%), and conflict markers and recipient richness move results by only a few points without a clear trend. Composition-hop queries outperform direct queries (64% vs. 43%), but this subset is restricted to easier settings on the other axes and we treat the gap as a selection artefact. Edit difficulty is driven primarily by competing in-context information, not by the structured complications the benchmark was designed to isolate.

Two generation-level effects sharpen the picture further. First, on composition-hop queries 9 of 35 correct cases match only via aliases rather than the literal target value, while every correct direct-query case contains the literal value. This suggests that the edit can support one-hop inference over the modified fact; because we do not run a null-edit baseline, we treat it as suggestive rather than conclusive. Second, edit success depends sharply on tokenization: single-token targets achieve 45% accuracy (585 cases), multi-token targets 0% (10 cases). The edit still strongly increases the probability of the first token on the multi-token subset (mean ≈0.75, often > 0.99), comparable to the single-token successful cases. The failure lies not in writing the direction but in completing it—the model frequently assigns high probability to the first target token yet generates a hedge or an adjacent attribute instead. This is the same first-token-P trap the metric section flags (§5.2).

Persistence is the sharpest failure mode. Figure 3 shows accuracy dropping from 45% at ft=0 to 21–23% at ft=1; additional turns have little further effect, with accuracy settling between 14% and 27%. Adversarial filler is marginally worse than neutral, but the dominant effect is the first-turn cliff: the state delta does not survive a single additional recurrent update of unrelated content.

Conditional on the edit landing, it is local. Figure 4 decomposes cases into the four-way joint outcome of edit and probe: the slice where the edit succeeds but an unrelated in-passage fact is

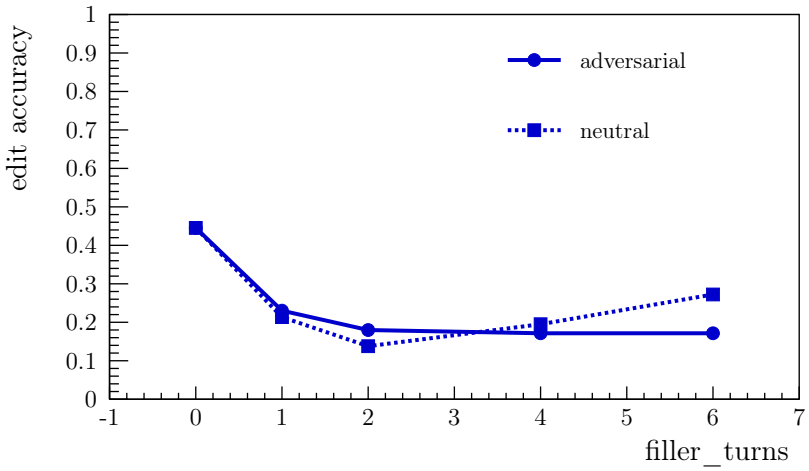

Figure 3: **Multi-turn filler decay.** X-axis: `filler_turns` $\in \{0, 1, 2, 4, 6\}$. One line per filler mode (neutral and adversarial); both share the ft=0 baseline. The dominant effect is the ft=0 $\rightarrow$ ft=1 drop; later turns do not materially worsen the outcome, and mode-dependence is small.

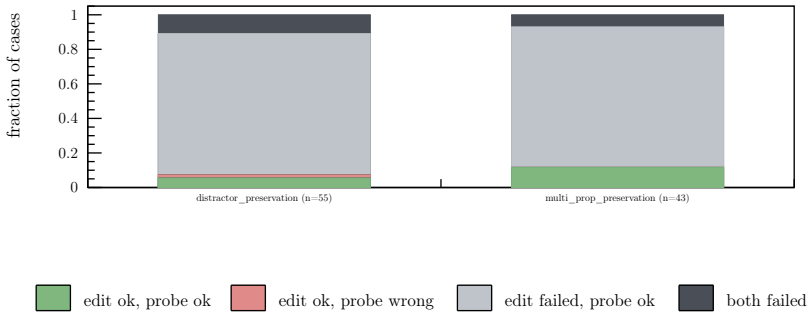

Figure 4: **Blast radius (joint edit $\times$ probe outcome).** Each case contributes an edit verdict (majority of its edit rows) and a probe verdict (majority of its probe rows) per probe kind. Stacked bars show the 4-way joint distribution. Green is clean success; coral is honest blast radius (edit landed, probe broken); grey shades are vacuous (edit never took). Across probe kinds the bled slice is small relative to the clean slice: when the edit lands, it is local.

corrupted is small relative to the clean-success slice. At the benchmark's fact-probe granularity the method does not systematically overwrite nearby facts; broader narrative spillover still occurs in richer contexts (§6), but locality is not the primary failure mode.

Taken together, the picture is consistent: the edit has a near-ceiling effect in the cleanest regime (92% with aligned calibration), a measurable but limited effect on average (45%), is driven mainly by competing context rather than the benchmark's structured complications, and fails to persist across even a single additional turn. The state delta appears to act as a transient perturbation that is cleanly localized but easily overwritten by subsequent computation.

5.6 Ablations

Table 2 collects the headline numbers across the conditions we swept on the test split. Main results (Figures 1–4) use the thinking off, ft=0 row as the baseline; Figure 3 adds the ft$\in \{1, 2, 4, 6\}$ rows for both neutral and adversarial filler. The thinking on row is a full re-run on the same test split; it lands identical to thinking off because RWKV's chat template ignores the `enable_thinking` flag for these checkpoints and the model emits a thinking scratch inline

condition	edit acc	clean slice	probe preservation
thinking off, ft=0 (headline)	45%	67%	92%
thinking on, ft=0	45%	67%	92%
thinking off, ft=1 neutral	21%	n/a	n/a
thinking off, ft=2 neutral	14%	n/a	n/a
thinking off, ft=4 neutral	19%	n/a	n/a
thinking off, ft=6 neutral	27%	n/a	n/a
thinking off, ft=1 adversarial	23%	n/a	n/a
thinking off, ft=2 adversarial	18%	n/a	n/a
thinking off, ft=4 adversarial	17%	n/a	n/a
thinking off, ft=6 adversarial	17%	n/a	n/a

Table 2: Headline edit accuracy across runtime conditions on the test split (595 edit rows per ft, 503 probe rows at ft=0). The clean ceiling is the mp=1∧ow=0∧dc=0 slice (n=21); only the ft=0 ceiling is reported — per-ft clean slices are too thin to separate signal from noise (on the same 21 cases the neutral sequence runs 14, 5, 0, 3, 2 correct across ft=0,1,2,4,6, with the zero-point driven by degenerate "Understood."-style completions rather than a distinct regime). "thinking on" leaves numbers unchanged: the chat template's `enable_thinking` flag does not gate the scratch on these RWKV checkpoints. All accuracy numbers use `correct_stripped` (§5.2).

regardless (we score against `correct_stripped` to avoid penalising the scratch).

This table is intentionally not presented as a set of benchmark wins. Its role is to anchor the workshop-level claim: the effect is visible, measurable, and cheap to apply, but not yet robust enough to support a "solved editing" framing.

Deferred ablations. Three component ablations—(i) calibration prompts wrapped in the chat template versus bare, (ii) editing the wkv component alone versus wkv jointly with the token-shift buffer, and (iii) varying the layer range over which the delta is applied—are straightforward to run with the existing pipeline but are deferred to a subsequent revision. The qualitative direction of each is already supported by the paper: the chat-template requirement is argued from the alignment constraint in §4; the wkv-only regime is shown to be nearly equivalent to wkv+token-shift by causal transplantation in Table 1; and the upper-half layer range is motivated by the same table's localization result. A systematic sweep remains future work.

6 Limitations

The limitations we observe are central to the paper's conclusions. While a simple state-delta operation can steer RWKV's recurrent memory, the resulting edits are fragile, only moderately reliable, and highly sensitive to context. When the edit lands, it is usually local at the fact-probe level; the larger problem is that it often fails to land or persist. We highlight several failure modes that illustrate these constraints.

Same-entity property crosstalk. Edits are not cleanly isolated to the target property. Modifying one attribute of an entity (e.g., hat color) can unintentionally affect other attributes of the same entity (e.g., eye color). This crosstalk is asymmetric: properties established earlier in the context are more resistant to modification than those introduced later. This suggests that the state representation entangles properties of the same entity rather than maintaining fully separable bindings.

Narrative spillover. In richer contexts, edits can generalize beyond the target entity or property. For example, changing 'yellow hat" to 'black hat" may also alter the color of other objects

in the scene. This indicates that the state delta often captures a broader semantic direction (e.g., a shift in color distribution) rather than a precise, slot-specific update.

Interference from parametric memory. Our method operates solely on the recurrent state and does not modify model weights. When parametric memory strongly encodes a fact about an entity, the edit must compete with this prior. In such cases, larger or more aggressive edits are required, which in turn increase the likelihood of unintended side effects.

Limited robustness under composition and time. As shown in Section 5, edit success degrades under realistic conditions: prior overwrites, multi-property edits, compositional queries, and intervening dialogue all reduce reliability. These results indicate that the edited state does not form a stable, compositional representation that can be consistently queried or maintained.

Implications. Taken together, these limitations suggest that RWKV's recurrent state supports a form of soft memory editing: it can be nudged in a desired direction, but does not provide precise, durable control over individual facts. We therefore view the proposed method not as a practical editing tool, but as a probe for understanding how factual information is represented and manipulated in recurrent working memory. Improving robustness and selectivity remains an open problem for future work.

7 Related Work

Knowledge editing in transformers. Prior work on knowledge editing focuses on modifying parametric memory. ROME [Meng et al., 2022] and MEMIT [Meng et al., 2023] compute targeted weight updates to MLP layers identified via causal tracing, while Mitchell et al. [2022] train hypernetworks to predict such edits. These methods produce persistent changes, require backpropagation, and affect all downstream uses of the model. Yao et al. [2023] provide a survey of this literature. In contrast, our approach operates on the per-session recurrent state, requires no gradients, and applies only to the current interaction.

In-context knowledge editing. A complementary line of work modifies model behavior through prompting rather than weight updates. For example, Zheng et al. [2023] show that providing in-context demonstrations of a fact change can steer model outputs, and Hernandez et al. [2024] study interventions on internal representations at inference time.

On RWKV, the distinction between in-context editing and our method is subtle but important. In-context edits are written into the recurrent state via the model's standard token-driven update rule, incurring additional context length and computation at each step. In contrast, our method directly computes a state delta from a calibration pair and applies it in a single operation. Both approaches ultimately modify the same underlying state, but differ in how the modification is produced. This shared substrate makes RWKV a natural setting for direct comparison, which we leave to future work.

State level intervention in State Space Models. State-level interventions on State Space Models have been studied for behavioral steering Sharma et al. [2024], where a direction is estimated from many contrastive pairs and added to the state to shift aggregate behavior. Our setting differs: we target a specific in-context binding using a single two-prompt calibration, and evaluate whether the model's answer to a specific query is rewritten. Weight-level knowledge editing has been studied on SSMs Paulo et al. [2024], producing persistent, cross-session edits via constrained optimization. Our method makes no weight changes and produces per-session edits only. These settings are complementary rather than competing.

Recurrent and state-space models. Recent architectures such as RWKV [Peng et al., 2023, 2024, 2025] and Mamba [Gu and Dao, 2023] maintain a persistent recurrent state that summarizes past inputs. This state provides a natural locus for runtime memory manipulation. To our knowledge, direct inference-time editing of factual associations via state intervention has not been previously demonstrated in these models. We adapt causal tracing techniques from Meng et al. [2022] to the recurrent setting (§3), using the state itself as the intervention target.

KV-cache editing. In transformer models, the KV cache is deterministically derived from the input sequence and is typically recomputed when the sequence is revisited. As a result, it is not a stable target for persistent editing. Existing approaches to session-level behavior modification therefore operate either through prompting [Zheng et al., 2023] or weight updates [Meng et al., 2022, 2023, Mitchell et al., 2022]. By contrast, RWKV's recurrent state is a fixed-size, persistent summary across tokens, making it a qualitatively different intervention surface.

Memory-augmented architectures. Explicit memory architectures—such as Memory Networks [Weston et al., 2015, Sukhbaatar et al., 2015] and Neural Turing Machines [Graves et al., 2014]—expose memory as a structured, trainable component. Our setting is complementary: rather than designing a model with an explicit memory interface, we investigate whether memory-like behavior that emerges in a standard recurrent model can be manipulated directly at inference time.

8 Conclusion

We report a simple empirical finding: in RWKV-7, in-context factual associations stored in recurrent working memory can be edited at inference time by adding a state delta derived from two short calibration prompts. The operation is a single vector subtraction and addition on the wkv state and residual stream, requires no gradients or weight updates, and takes effect immediately.

On the RIME test split the edit succeeds on 45% of cases while preserving 92% of unrelated in-passage facts. Read together, these two numbers—and the four figures of §5.5—tell a more specific story than either "the method is broadly robust" or "the method simply fails." The edit carries a real signal and, when it lands, it is usually well-localized to the intended binding. But the mechanism is still brittle overall: success remains far from reliable, the difficulty surface is dominated by competing in-context material (especially distractors), and the edit does not survive a single intervening filler turn. In other words, the main limitation exposed by the benchmark is not blast radius but lack of robustness and durability.

We read this as a claim about how information is written to RWKV's recurrent state, not about how it is stored. A state delta estimated from two calibration prompts captures the right direction to rewrite the target fact, but that direction is not stably re-encoded by subsequent recurrent updates; the next write effectively overwrites it. In this view, RWKV's state supports a usable but transient runtime edit—useful for one-shot interventions, not for multi-turn dialogue.

Concretely, the results argue against two natural stronger readings of the mechanism: (i) that the delta is a generic semantic nudge (locality is good, not bad); and (ii) that failure modes are cleanly indexed by the benchmark's "harder" axes (the difficulty surface is flatter than designed, and distractors dominate it). They also argue against the weaker reading that the method is broken: the edit carries a real, measurable signal that is already cheap enough to apply per-session.

The central takeaway is therefore not that runtime memory editing is solved, but that there is a simple, reproducible, and cheap editing mechanism that can be used as a probe of recurrent working memory. Future work should ask why the edit direction is not durable: whether the decay we observe is attributable to the time-mix decay term w_t, to rank-1 interference from

subsequent tokens, or to the calibration delta being off-manifold in a way that the next update quickly renormalizes. Improving durability—so that an edited state remains coherent across multi-turn dialogue—is the first open problem we would prioritize.

References

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. Inspecting and editing knowledge representations in language models. In *Conference on Language Modeling (COLM)*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=0DcZxeWfOPt`.

Gonçalo Paulo, Thomas Marshall, and Nora Belrose. Does transformer interpretability transfer to rnns?, 2024. URL `https://arxiv.org/abs/2404.05971`.

Bo Peng, Eric Alcaide, Quentin Anthony, et al. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936. URL `https://aclanthology.org/2023.findings-emnlp.936/`.

Bo Peng, Daniel Goldstein, Quentin Anthony, et al. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Bo Peng, Daniel Goldstein, Quentin Anthony, et al. RWKV-7 "goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL `https://aclanthology.org/D19-1250/`.

Arnab Sen Sharma, David Atkinson, and David Bau. Locating and editing factual associations in mamba, 2024. URL `https://arxiv.org/abs/2404.03646`.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, 2015.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.632. URL https://aclanthology.org/2023.emnlp-main.632/.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

A RIME dataset format

Each RIME case is a JSON record centred on a single fictional person and a single attribute drawn from a small closed-vocabulary set (e.g. hat color, city, pet, vehicle). The record bundles everything needed to construct the recipient state, apply an edit, and score the outcome.

At a high level, a case contains:

- a **target binding**: the entity, the property, the pre-edit value (`value_old`) and the post-edit value (`value_new`);
- a **recipient**: one or more LLM-written passages that establish `value_old` in context, together with an `entity_order` and `target_position` that record where the target is mentioned relative to distractors;
- one or more **edits**, each a `source`/`dst` pair written in a different scene from the recipient (so the edit cannot rely on surface overlap);
- a list of **queries** (paraphrases of the target question) and **probe queries** about unrelated facts in the same passage, used to measure blast radius;
- a **parameter vector** recording the case's position on the RIME control axes (distractor count, overwrite count, composition hop, multi-property count, stacking depth, etc.).

A simplified example (one vehicle case, shortened for display):

```
{
  "entity":    "Rohan",
  "property":  "vehicle",
  "value_old": "hatchback",
  "value_new": "minivan",

  "recipient": {
    "passages": [
      "Rohan drives a hatchback while Esme rides a scooter,
       Amara rides a motorcycle, Soren drives a convertible,
       and Noor drives a sedan."
    ],
    "entity_order":    ["Rohan","Soren","Amara","Noor","Esme"],
    "target_position": 0
  },

  "edits": [
    {"style":  "canonical_bare",
     "source": "Rohan's vehicle is hatchback.",
     "dst":    "Rohan's vehicle is minivan."}
  ],
```

```
  "queries": [
    "What type of vehicle does Rohan drive?",
    "What is Rohan's vehicle?"
  ],
  "expected_answer_substring": "minivan",

  "probe_queries": [
    {"query": "What is Esme's vehicle?",  "expected": "scooter"},
    {"query": "What is Amara's vehicle?", "expected": "motorcycle"}
  ],

  "params": {
    "distractor_count": 4,
    "overwrite_count":  0,
    "composition_hop":  false,
    "multi_prop_count": 1,
    "stack_depth":      1
  }
}
```


A case is retained only if the unedited model correctly answers the target query from the recipient passage, so failures reported in §5.5 reflect the editing mechanism rather than passage ambiguity.