

CHRONOEXPERT: A Temporal Mixture-of-Experts for Knowledge Editing in Large Language Models

Xudong Li^{1,*} Yizhou Jin^{1,*} Dandan Song^{1,†} Yongyu Huo¹ Yuhang Tian¹
Jun Yang¹ Chenhao Li¹ Huipeng Ma¹ Luan Zhang¹ Xing Pei¹

¹Beijing Institute of Technology
demmon718@gmail.com, sdd@bit.edu.cn

Abstract

Knowledge editing for large language models typically treats a fact update as a replacement in which a new object overwrites the old. Many real-world facts, however, evolve through sequences: the same subject–relation pair maps to different objects across disjoint intervals, each correct within its validity window. Existing temporal-editing benchmarks probe latest-value or explicit-time recall, so they cannot determine whether an edited model retains a coherent timeline. We introduce TKE-BENCH, a diagnostic benchmark that operationalizes *temporal accessibility* along three reasoning-oriented axes with paraphrased variants. On TKE-BENCH, all evaluated editing baselines collapse on chronologically structured queries, revealing *catastrophic temporal interference* (CTI). We trace CTI to a geometric property implicit in locate-and-edit methods: mid-layer subject-token keys are nearly collinear across time slices of the same subject, so stacked updates write along one direction and the latest edit overwrites every earlier value. Building on this, we propose CHRONOEXPERT, a temporal mixture-of-experts that stores each time slice in an independent lightweight expert and activates only the matching one at inference, leaving the base model otherwise untouched. Across multiple LLM backbones, CHRONOEXPERT consistently outperforms the strongest editing baselines and stays robust as chains grow longer.

1 Introduction

Large language models (LLMs) (Zhao et al., 2023; Naveed et al., 2025) store large amounts of factual knowledge in their parameters (Dai et al., 2022), but this knowledge becomes outdated as the world changes, causing hallucinations and time-inconsistent predictions (Huang et al., 2025; Dhingra et al., 2022). *Knowledge editing* (KE) (De Cao

et al., 2021; Meng et al., 2022a,b; Yao et al., 2023; Wang et al., 2024c) revises a deployed model in place, shifting its prediction for a given subject and relation from an outdated object to a desired one, and recent locate-and-edit advances refine where and how the update lands (Fang et al., 2025; Xu et al., 2025; Gupta et al., 2025). Yet most KE methods treat each edit as a one-shot replacement that overwrites the old fact. Many real-world facts, however, evolve through a sequence of values: the same subject and relation map to different objects across non-overlapping intervals, and every earlier object stays correct within its own interval. LeBron James, for instance, played for the Cleveland Cavaliers, the Miami Heat, the Cavaliers again, and now the Los Angeles Lakers, all four valid under their respective intervals. ATOKe (Yin et al., 2024) formalizes this as *temporal knowledge editing* (TKE), where an editor must incorporate emerging facts while preserving historical ones.

Whether an editor truly supports such knowledge cannot be told from current or explicit-time queries alone. Existing TKE benchmarks (Yin et al., 2024) focus on current and explicit-time recall, while broader KE benchmarks (Chen et al., 2025) evaluate non-temporal edit effects; neither reveals whether the edited facts form a coherent timeline reachable through diverse temporal reasoning. We therefore introduce TKE-BENCH, a diagnostic benchmark that operationalizes *temporal accessibility* along three reasoning-oriented axes: temporal state retrieval, entity-anchored *before/after* navigation, and reverse temporal localization, each with paraphrased variants.

TKE-BENCH does not merely reveal that existing editors fail; it makes the failure measurable across distinct forms of temporal access. The failure has a simple geometric origin implicit in locate-and-edit methods, which stack rank-one updates through subject-token keys under a near-orthogonality assumption that holds across unre-

*Equal contribution.

†Corresponding author.

lated edits but breaks within a temporal chain. Subject-token activations at mid-layers encode *who* the subject is, not *when*, so different time slices of the same subject induce nearly parallel keys, a geometric property we later term *temporal key collinearity* (§4.2): each new update is written along almost the same direction as the previous one, and the latest edit overwrites every earlier value on the chain. We call this failure mode *catastrophic temporal interference* (CTI). On TKE-BENCH, every editing baseline we evaluate collapses on chronologically structured queries, confirming this geometric prediction.

Since CTI arises from parameter sharing along the subject-token direction rather than from the facts themselves, the remedy is to stop sharing parameters across time slices. We propose CHRONOEXPERT (Fig. 2), which casts TKE as a *mixture-of-experts* (Shazeer et al., 2017). The base model stays frozen. For each time slice we train a small LoRA-style (Hu et al., 2022) expert on the MLP down-projection of a mid-layer, storing that object–interval association in its own disjoint parameter slot. At inference, a lightweight router selects the expert whose time slice the query refers to; it is attached to the frozen backbone via a forward hook for generation, then removed, leaving the base model byte-identical to its pre-edit state. Because experts inhabit disjoint slots, time slices do not overwrite one another, and CHRONOEXPERT prevents the shared-parameter source of CTI by construction.

We make three contributions.

1. **Benchmark.** TKE-BENCH, a diagnostic TKE benchmark operationalizing *temporal accessibility* along three reasoning-oriented axes with paraphrased variants.
2. **Diagnosis.** We identify and name *catastrophic temporal interference* (CTI), tracing it geometrically to near-parallel keys across same-subject time slices, a root cause invisible to prior benchmarks.
3. **Method.** CHRONOEXPERT, an MoE-style per-time-slice expert framework that prevents CTI by construction, with consistent gains over editing baselines across multiple LLM backbones.

2 Related Work

Knowledge editing (De Cao et al., 2021; Yao et al., 2023; Wang et al., 2024b) updates deployed mod-

els without retraining. Parameter-editing methods modify weights directly: ROME (Meng et al., 2022a) applies a rank-one update to MLP factual associations, MEMIT (Meng et al., 2022b) batches many such updates, AlphaEdit (Fang et al., 2025) projects perturbations onto the null space of preserved knowledge for sequential editing, and MEND (Mitchell et al., 2021) learns a fast gradient transformation. Parameter-preserving methods (SERAC (Mitchell et al., 2022), IKE (Zheng et al., 2023), WISE (Wang et al., 2024a), retrieval-based editors) keep the backbone frozen and condition generation on stored edits. Both adopt a *replacement view* in which a new object overwrites the old, which does not fit time-evolving facts where multiple objects stay correct under different validity intervals.

Temporal knowledge editing (TKE) targets this setting directly. ATOKe (Yin et al., 2024) formalizes it as temporal fact chains and probes current and historical values under explicit or relative queries, and METO (Yin et al., 2024) adds a multi-editing time objective; on these benchmarks conventional editors preserve the latest edit at the expense of earlier ones. We instead evaluate *temporal accessibility*: whether edited facts form a coherent timeline reachable through state retrieval, entity-anchored before/after navigation, and reverse temporal localization. This evaluation makes measurable a failure mode we call *catastrophic temporal interference* (CTI): same-subject edits overwrite earlier time slices along a shared subject-token direction. We address this failure with CHRONOEXPERT, which assigns each time slice to a lightweight expert rather than storing all temporal values in shared parameters.

3 TKE-BENCH

TKE-BENCH is a diagnostic benchmark that tests whether edited temporal facts remain accessible as a timeline. Unlike evaluations that ask only whether the latest value is installed, or whether a value can be recalled from an explicit year, TKE-BENCH asks whether each time slice stays reachable under multiple anchors: the present time, a calendar interval, a neighboring entity, or the entity itself, whose validity interval must be recovered.

Subject: LeBron James Relation: <playsFor>			
Chain: $\langle (\text{Cleveland Cavaliers, 2003–2010}), (\text{Miami Heat, 2010–2014}), (\text{Cleveland Cavaliers, 2014–2018}), (\text{Los Angeles Lakers, 2018–present}) \rangle$			
qtype	Question (orig.)	Question (para.)	Gold answer
current	Which team does LeBron James currently play for?	What team is LeBron James a member of these days?	Los Angeles Lakers
simple_entity	Which team did LeBron James play for from 2010 to 2014?	Which club was LeBron James with between 2010 and 2014?	Miami Heat
before	Which team did LeBron James play for before the Heat?	What team did LeBron play for prior to joining the Heat?	Cleveland Cavaliers (2003–2010)
after	Which team did LeBron James play for after the Heat?	Where did LeBron James go right after leaving the Heat?	Cleveland Cavaliers (2014–2018)
simple_time	When did LeBron James play for the Miami Heat?	In which years was LeBron James a Miami Heat player?	2010–2014

Table 1: A TKE-BENCH instance on one (s, r) chain (five question types $\times$ two subtypes). before/after annotate the matched interval in parentheses, since *Cleveland Cavaliers* recurs in this chain.

Capability	ATOKE	TKE-BENCH
Current fact query	✓	✓
Explicit-time entity query	✓	✓
Entity $\rightarrow$ time query	✗	✓
Before/after entity anchors	✗	✓
Per-question paraphrase	✗	✓

Table 2: Coverage comparison between ATOKE and TKE-BENCH.

3.1 Task Formulation and Question Taxonomy

We represent a temporal fact as a five-tuple (s, r, o, t_s, t_u) , where s is a subject, r a relation, o an object, and $[t_s, t_u)$ the half-open interval over which o is the valid answer to (s, r) . For a fixed (s, r) pair, the ground truth over time is a chain

$$\mathcal{C} = \langle (s, r, o_i, t_{s_i}, t_{u_i}) \rangle_{i=1}^n, \quad t_{u_i} \leq t_{s_{i+1}}, \quad (1)$$

where the time slices are temporally disjoint. Given a base model $\mathcal{M}$ and a temporal chain $\mathcal{C}$, temporal knowledge editing learns an editor F that returns an edited model $\mathcal{M}_e = F(\mathcal{M}, \mathcal{C})$. The edited model should answer any query grounded in $\mathcal{C}$ with the object or time interval referred to by that query, rather than collapsing the chain onto a single most recent value.

We operationalize this objective as *temporal accessibility*: whether the edited facts can still be reached through different forms of temporal access, realized as five question types along three axes. **State retrieval** asks which object holds now or during a stated interval: current queries the present object o_n , and simple_entity returns o_i given its validity interval $[t_{s_i}, t_{u_i})$. **Entity-anchored navigation** asks for the predecessor or successor of a known chain entity: before and after query the neighboring objects o_{i-1} and o_{i+1} given an anchor entity o_i . **Reverse temporal localization**

asks when a given object was valid: simple_time queries the interval $[t_{s_i}, t_{u_i})$ given o_i .

Crucially, temporal accessibility goes beyond explicit-time recall: before and after require the chain to be navigable from an entity anchor, and simple_time reverses the direction of access, recovering the temporal support of a known entity rather than an entity from time.

Orthogonally to these question types, every base question is duplicated into two subtypes. The original subtype is the raw template output, and the paraphrased subtype is a GPT-4o (Hurst et al., 2024) rewrite that preserves the gold answer. This pair isolates surface-form robustness from the underlying reasoning axis. Table 1 expands one chain into all five question types and both subtypes; we report results for every (question type, subtype) cell.

3.2 Construction

We build TKE-BENCH on ATOKE-ME, the multi-editing split of ATOKE, whose (s, r, o, t_s, t_u) tuples derive from YAGO 3.0.3 (Suchanek et al., 2007). ATOKE extracts time-scoped triples across nine relations, keeps only facts a GPT-J probe can recall, and augments each chain with synthetic future-dated facts so a strictly newer edit is always available; its multi-editing split frames every fact along a chain as a successive edit rather than a single old-to-new replacement. We re-aggregate the records by (s, r) and sort each group by start time, recovering 8,819 chains over 27,189 facts, each passed through the question-generation procedure below.

Base questions come from manually written templates per relation and question type. For each time slice $(s, r, o_i, t_{s_i}, t_{u_i})$, the templates instantiate current, simple_entity, and simple_time,

plus before/after except at chain endpoints, where no gold neighbor exists (before for the first slice, after for the last).

We use GPT-4o to generate the paraphrased variant of each base question, under a prompt that preserves the gold answer, followed by lightweight format normalization and non-empty field checks. Two graduate-student annotators then independently audit a 10% stratified sample of TKE-BENCH-1K, and more than 95% of items are judged faithful by both. Full generation prompts and the annotation protocol are provided in Appendix E.

Throughout the paper, all experiments are conducted on TKE-BENCH-1K, a curated subset of this pool. We first restrict the pool to chains of length $n \in [2, 7]$, since length-1 chains carry no temporal conflict and are not eligible. We then draw 1,000 chains by relation-stratified random sampling with seed 42, so that each relation’s share mirrors its share in the underlying pool, yielding 2,982 temporal facts. TKE-BENCH-1K is the curated set on which all numbers in this paper are measured.

3.3 Relation to ATOKE

TKE-BENCH builds on ATOKE rather than replacing it: reusing its fact backbone ensures that evaluation differences come from the query interface, not from a new source of facts. Whereas ATOKE mainly checks whether edited values are installed and recoverable under explicit temporal cues, TKE-BENCH additionally probes entity-anchored access, reverse temporal localization, and paraphrased variants (Table 2).

3.4 Dataset Summary

After endpoint filtering, the 1,000 chains of TKE-BENCH-1K yield 21,856 questions across the 10 question-type/subtype cells, with mean chain length 2.98. We report the full per-type counts in Appendix A.2, and the chain-length and per-relation distributions, which follow the ATOKE-ME pool, in Appendix A.3.

3.5 Evaluation Protocol

Unless otherwise specified, we evaluate on all of TKE-BENCH-1K. For entity-valued questions (current, before, after, simple_entity), we use bidirectional substring matching between generation and gold entity; for simple_time, a prediction is correct if both endpoint years, or the unique

year for a single-year interval, appear in the generation. The same rules apply to all editors and baselines.

We report accuracy at three granularities: the individual (question-type, subtype) cell; per-question-type scores, averaged over the two subtypes; and an overall micro-average over all questions. Per-relation analyses appear in Appendix A.3, and per-chain-length analyses in §6 and Appendix C.3.

4 Diagnosis: Anatomy of Catastrophic Temporal Interference

We characterize *catastrophic temporal interference* (CTI), the failure mode introduced in §1. On TKE-BENCH, the editing baselines we evaluate collapse on chronologically structured queries; for locate-and-edit methods, the collapse traces to a single geometric property: same-subject keys across time slices are nearly parallel, so stacked rank-one updates project onto one another and the latest edit overwrites every earlier value on the chain. We call this property *Temporal Key Collinearity* (TKC). It is invisible to prior benchmarks that probe only a single time slice, and dictates the shape of the remedy in §5.

4.1 Failure Profile on Chronological Queries

The collapse follows a consistent profile across editors and backbones; we sketch its qualitative shape here and defer full numbers and chain-length breakdowns to §6. Stacking rank-one updates installs the most recent time slice (current) reasonably often but flattens every other question type, with the entity-anchored before queries collapsing the hardest. Adding an explicit time objective (METO) lifts simple_time in some configurations but leaves the other question types unchanged. The chain thus collapses onto its final slice: earlier values are functionally erased, though they were written into the weights only seconds before.

4.2 Temporal Key Collinearity

A ROME-style editor writes

$$\Delta W_i = \frac{(v_i^* - Wk_i) k_i^\top}{\|k_i\|^2}, \quad (2)$$

where $k_i \in \mathbb{R}^d$ is the mid-layer activation at the subject token of time slice i and v_i^* is the target value. Applying the stacked update $\Delta W = \sum_{i=1}^n \Delta W_i$

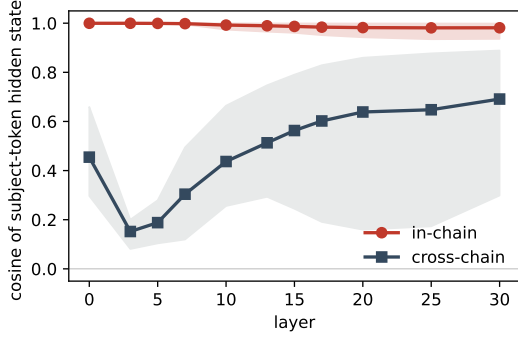


Figure 1: Temporal Key Collinearity on LLaMA-2-7B (§4.2): per-layer cosine similarity of subject-token hidden states for in-chain time slices vs. random cross-chain pairs, over all 1,000 chains of TKE-BENCH-1K.

to a query key k_j yields

$$\Delta W k_j = \sum_{i=1}^n (v_i^* - W k_i) \frac{\langle k_i, k_j \rangle}{\|k_i\|^2}. \quad (3)$$

When key norms are comparable, this coefficient is governed by the cosine similarity between k_i and k_j . Across unrelated subjects this coefficient is small, so each edit is recovered independently. This is the standard near-orthogonality assumption under which ROME, MEMIT, and AlphaEdit operate. Within a temporal chain the assumption breaks. On LLaMA-2-7B and all 1,000 chains of TKE-BENCH-1K, we measure the cosine similarity of subject-token hidden states across in-chain slices and across random pairs from different chains (Fig. 1). In-chain similarity stays above 0.98 at every measured layer, far above the cross-chain baseline at the edit-relevant layers, with the gap largest exactly where locate-and-edit methods write (Appendix D.1). Because the in-chain coefficients are this close to one, every time slice’s target v_i^* contributes almost identically to $\Delta W k_j$, and sequential edits let the latest target dominate along the shared direction. The left panel of Fig. 2 traces this collapse on the running chain: the four time slices induce near-collinear keys $k_1, \dots, k_4$ ($\cos \approx 1$), so their rank-one updates $\Delta W_1, \dots, \Delta W_4$ accumulate in a single shared editor in which the most recent edit dominates and the earlier slices are overwritten.

Persistence under stronger editors. MEMIT solves a multi-edit normal equation $\Delta W K = R$ in closed form, where $K = [k_1 \dots k_n]$ stacks all keys and R stacks residuals. Near-parallel K is rank-deficient, so the linear system cannot separately fit

v_i^* for each k_i , and the regularized solution averages targets along the shared direction. AlphaEdit further constrains the update to lie in the null space of a preserved-knowledge covariance, which prevents interference with unrelated facts but does not change the within-chain rank deficiency. METO adds a time embedding to v_i^* , modifying targets without altering k_i , so the keys remain parallel and overwriting is delayed rather than prevented. Any editor whose chain edits share parameters along an axis indexed by the subject token is geometrically predisposed to CTI.

Why subject-token keys are parallel. Subject-token activations at a mid-layer encode who the subject is, not when. Time information lives in the calendar tokens of the prompt and is only fused at the last token in later layers, so the subject token is largely invariant under temporal slicing of the same chain. TKC is therefore a structural property of the representation rather than an artifact of any particular editor. This suggests that changing the layer, rank, or optimizer alone is unlikely to remove TKC unless the representation used for editing also changes.

5 CHRONOEXPERT: A Temporal Mixture-of-Experts

5.1 Formulation

§4 established that any editor writing multiple chain edits into shared parameters along the subject-token direction is geometrically predisposed to CTI. CHRONOEXPERT (Fig. 2, right) prevents the shared-parameter source of CTI by construction: it stores each time slice in a disjoint parameter slot and activates exactly one slot at inference, so that chain edits are not written into the same editable parameter slot. Throughout this section the base model $\mathcal{M}$ is read-only; the only learnable objects are the time slice experts.

Given a chain $\mathcal{C} = \{(s, r, o_i, t_{s_i}, t_{u_i})\}_{i=1}^n$, we model the edited generator as a hard-routed mixture of n experts over $\mathcal{M}$,

$$P_{\mathcal{M}_e}(y \mid q, \mathcal{C}) = P_{\mathcal{M} + E_{\pi(q, \mathcal{C})}}(y \mid q). \quad (4)$$

Each E_j is a parameter perturbation that is added to $\mathcal{M}$ only when the router $\pi(q, \mathcal{C})$ selects time slice j ; unselected experts stay inactive. We route among one chain’s n slices (the single-chain setting), deferring out-of-chain abstention to lifelong editing. The router π is the maximum a posteriori approximation of the time slice posterior $p(j \mid q, \mathcal{C})$, and

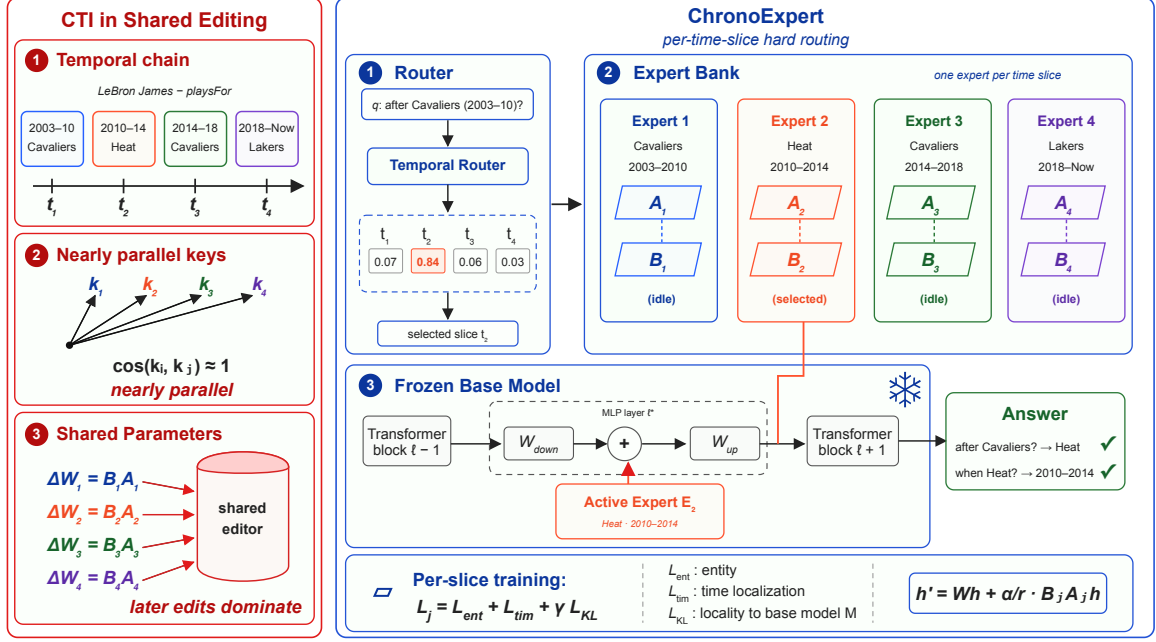


Figure 2: Overview of CHRONOEXPERT on the running chain *LeBron James-<playsFor>*. **Left**: the catastrophic temporal interference (CTI) of shared editing, diagnosed in §4. **Right**: CHRONOEXPERT gives each time slice its own expert and routes each query to a single one (here slice t_2).

is treated as a stand-alone module whose concrete realizations are described in §6.

The routing axis is what distinguishes CHRONOEXPERT from prior MoE-style (Shazeer et al., 2017) editors. Memory-based editors such as WISE (Wang et al., 2024a) route by content semantics between a base store and a side store; here the routing axis is *time*, which matches the structure of TKE in which each query belongs to exactly one time slice. The same structure also dictates hard rather than soft routing. A soft mixture would blend experts that are meant to encode mutually exclusive intervals, reintroducing cross-slice interference.

5.2 Slice Experts

We instantiate each E_j as a low-rank additive perturbation (Hu et al., 2022) on one mid-layer MLP (the Expert Bank of Fig. 2). Following the locate-and-edit literature (Meng et al., 2022a; Dai et al., 2022; Meng et al., 2022b), the modified module is an MLP down-projection, the family these methods edit for (s, r) recall, at a per-backbone mid-layer ℓ^* not tied to a fixed causal-tracing locus. For an input h at the down-projection,

$$h' = Wh + \frac{\alpha}{r} B_j A_j h, \quad (5)$$

with low-rank factors $A_j \in \mathbb{R}^{r \times d_{\text{tr}}}$ and $B_j \in \mathbb{R}^{d \times r}$ of rank r and scaling α . Only A_j and B_j are learned; the base weight W is frozen and shared across all time slices of all chains. Freezing W has two consequences. First, with no expert attached the model is exactly $\mathcal{M}$, so any knowledge the active chain does not touch is preserved. Second, chains served in sequence cannot contaminate one another, because removing the hook for E_j returns the model to byte-identical pre-edit state.

The edit layer ℓ^* is fixed per backbone; the per-backbone layers, rank r , and scaling α are reported in Appendix B.2.

5.3 Training Objective

Each expert is trained on its own time slice $(s, r, o_j, t_{s_j}, t_{u_j})$ in isolation. No information flows between experts and no joint optimization is performed, so the per-chain training cost grows linearly in the chain length n . The objective combines a bidirectional next-token loss with a locality regularizer,

$$\mathcal{L}_j = \lambda(\mathcal{L}_j^{\text{ent}} + \mathcal{L}_j^{\text{tim}}) + \gamma \mathcal{L}_j^{\text{KL}}, \quad (6)$$

with task weight λ and locality weight γ .

The two task-side terms train the expert to recover the time slice in both directions of the

(object, time) pair. $\mathcal{L}_j^{\text{ent}}$ is the next-token cross-entropy for predicting the entity o_j from a time-prefixed prompt of the form “From t_{s_j} to t_{u_j} , $r(s) = ?$ ”, covering the current and simple_entity question families. $\mathcal{L}_j^{\text{tim}}$ is the next-token cross-entropy for predicting the year range $[t_{s_j}, t_{u_j})$ from a simple_time-style question in which the entity o_j is given. Training both directions makes each expert encode both the entity and its validity interval. For before/after, the router maps the anchor entity to the neighboring slice, and the selected expert then generates that slice’s entity.

The locality term $\mathcal{L}_j^{\text{KL}}$ is the forward-direction KL divergence between $\mathcal{M} + E_j$ and $\mathcal{M}$, evaluated on a fixed pool of locality probes drawn from chains disjoint from $\mathcal{C}$. This term penalizes the expert for changing the base model’s output distribution on knowledge that the time slice does not concern, mirroring the standard locality loss of locate-and-edit editors. We train each expert with Adam; the loss weights, learning rate, and epoch budget are given in Appendix B.2.

5.4 Inference

Inference for a single query q against a chain $\mathcal{C}$ proceeds in three steps. The router π first scores all n slots of $\mathcal{C}$ given q and returns an index $\hat{j} = \arg \max_j \pi_j(q, \mathcal{C})$. The selected expert $E_{\hat{j}}$ is then attached to $\mathcal{M}$ via a forward hook on the layer- ℓ^* down-projection that adds $\frac{\alpha}{r} B_{\hat{j}} A_{\hat{j}} h$ to the projection output (Eq. 5). After generation completes, the hook is removed and the model returns to its base configuration. The right panel of Fig. 2 instantiates these three steps on the running chain: the query *after Cavaliers?* concentrates the router’s mass on slice t_2 , so only expert E_2 (Heat, 2010–2014) is attached while the other three experts stay idle, and the edited model answers both the entity query *after Cavaliers?* $\rightarrow$ Heat and the time query *when Heat?* $\rightarrow$ 2010–2014. No parameter of $\mathcal{M}$ is mutated, and the chain $\mathcal{C}$ can be deleted by discarding its n experts; a different chain is served by swapping in its own experts.

Because routing is the only learned component that depends on q at inference time, the end-to-end error of CHRONOEXPERT decomposes cleanly into editing error (how well the experts encode each time slice) and routing error (how often π picks the correct time slice). We exploit this decomposition in §6, where we compare three realizations of π : a heuristic over year and entity matches, a discriminative multiple-choice classifier, and a generative

log-likelihood scorer.

6 Experiments

6.1 Experiment Setup

Evaluation. We evaluate on TKE-BENCH-1K (§3) and report accuracy per (question type, subtype) cell, using the matching protocol of §3.5.

Baselines. We compare with ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), ALPHAEDIT (Fang et al., 2025), their METO-augmented variants (Yin et al., 2024), and WISE (Wang et al., 2024a). We use GPT-J-6B (Wang and Komatsuzaki, 2021), LLaMA-2-7B (Touvron et al., 2023) and LLaMA-3.1-8B-Instruct (AI@Meta, 2024) as the backbone LLMs in our main experiments. Detailed descriptions of all baselines are provided in Appendix B.1.

Experimental Setup. Each CHRONOEXPERT expert is a LoRA (Hu et al., 2022) adapter on the MLP down-projection at a per-backbone edit layer ℓ^* (Table 10), and the router is a Qwen2.5-0.5B (Team, 2024) generative scorer fine-tuned for slice selection on chains outside TKE-BENCH-1K. Additional experimental details, including router training data and LoRA hyperparameters, are provided in Appendix B.

6.2 Main Results

Table 3 shows the results on TKE-BENCH-1K. CHRONOEXPERT achieves the best Overall accuracy on every backbone, exceeding 85% on all three. Unlike baselines whose scores concentrate on current, CHRONOEXPERT improves all five question types, with the largest gains on the chronological-access queries before, after, and simple_time, where existing editors collapse along the CTI profile diagnosed in §4. On current, CHRONOEXPERT stays competitive with the strongest baseline, so temporal accessibility is recovered without sacrificing latest-fact editing. Finally, the **Org** and **Para** columns track closely across backbones, indicating that the gains are not tied to template wording or to a single model family.

6.3 Ablation and Analysis

Per-time-slice isolation. To isolate the contribution of per-time-slice parameter separation, we replace the experts with a single *Shared-LoRA*

Methods	current		after		before		simple_entity		simple_time		Overall
	Org	Para	Org	Para	Org	Para	Org	Para	Org	Para	
GPT-J-6B											
WISE	1.84	2.01	1.01	1.31	1.11	2.72	1.68	2.48	4.53	2.31	2.19
ROME	86.60	81.20	32.71	31.45	1.72	1.66	26.53	25.55	2.88	3.62	21.80
ROME+METO	93.30	85.70	36.44	32.61	1.97	1.56	28.54	26.83	14.19	15.66	26.40
MEMIT	94.40	91.40	41.02	41.02	1.01	0.96	32.33	30.65	3.09	3.45	25.61
MEMIT+METO	94.80	91.70	40.82	41.02	1.41	1.31	31.42	30.42	16.57	18.98	29.49
ALPHAEDIT	95.10	90.40	39.00	39.00	1.26	1.16	32.09	30.92	3.29	4.16	25.40
ALPHAEDIT+METO	89.70	83.40	35.78	35.13	1.56	1.56	30.72	28.20	14.05	13.15	26.39
CHRONOEXPERT (Ours)	94.90	92.20	73.73	86.56	75.43	77.70	89.07	89.34	95.41	95.37	87.35
LLaMA-2-7B											
WISE	21.93	18.81	19.83	15.55	7.37	7.87	15.46	13.08	17.34	15.02	15.63
ROME	87.50	84.00	39.46	40.01	1.87	1.97	28.20	28.24	5.37	7.04	24.80
ROME+METO	83.30	77.30	38.60	37.70	3.63	4.44	27.93	26.39	14.02	16.70	26.61
MEMIT	77.90	76.30	35.33	34.93	1.72	1.51	24.35	25.22	3.82	3.99	21.55
MEMIT+METO	74.70	73.10	35.33	36.79	3.58	4.24	25.05	25.59	14.35	15.69	25.03
ALPHAEDIT	92.50	84.20	40.97	37.29	2.07	1.77	32.06	31.05	11.27	13.68	27.55
ALPHAEDIT+METO	91.70	82.80	41.22	37.44	1.92	2.32	31.76	30.05	31.69	32.33	32.67
CHRONOEXPERT (Ours)	89.10	86.90	77.71	78.26	77.35	74.87	87.22	87.83	92.66	96.48	85.69
LLaMA-3.1-8B-Instruct											
WISE	18.98	19.72	36.49	32.66	1.82	1.82	19.45	18.68	16.00	17.24	18.30
ROME	64.50	66.30	36.03	33.62	4.79	4.49	25.49	24.58	5.57	5.63	21.51
ROME+METO	53.30	55.00	32.21	31.30	6.86	6.91	21.86	22.20	6.61	9.22	20.14
MEMIT	68.70	72.10	41.47	40.26	6.16	6.91	27.40	28.54	3.92	3.96	23.75
MEMIT+METO	53.50	53.10	30.95	31.25	6.51	7.37	22.27	22.30	8.69	9.49	20.34
ALPHAEDIT	39.80	43.10	22.95	20.53	8.98	8.83	21.19	22.27	19.62	16.63	20.23
ALPHAEDIT+METO	32.80	30.70	17.46	15.25	7.16	7.62	17.10	17.61	45.14	44.40	24.16
CHRONOEXPERT (Ours)	89.80	89.40	82.13	82.79	82.49	83.20	91.01	90.64	97.12	97.42	89.50

Table 3: Main results on TKE-BENCH-1K (accuracy %). Best per column in **bold**; shaded row ours.

Variant	GPT-J	LLaMA-2-7B	LLaMA-3.1-8B-Instruct
Shared-LoRA	32.15	42.34	44.48
CHRONOEXPERT	87.35	85.69	89.50
Δ	+55.20	+43.35	+45.02

Table 4: Per-time-slice isolation ablation in Overall accuracy (%).

adapter, holding the edit layer, rank, training objective, and router fixed across the two conditions (Table 4). Collapsing all chain edits into one shared adapter lowers Overall by more than 40 percentage points on every backbone, confirming the value of per-time-slice parameter isolation (Appendix C.1).

Router. We compare three realizations of π on LLaMA-2-7B (implementations in Appendix B.3): a rule-based *Heuristic*, a discriminative *DeBERTa-MC* classifier (He et al., 2021), and the default *Qwen-Gen* generative scorer. Table 5 shows that routing quality directly controls end-to-end performance: Qwen-Gen performs best, DeBERTa-MC closes much of the gap, and the Heuristic fails on entity-anchored before/after queries that carry no explicit calendar cue (Appendix C.2).

CHRONOEXPERT stays robust as chains

Router	Overall
Heuristic	45.37
DeBERTa-MC	78.70
Qwen-Gen	85.69

Table 5: Router ablation on LLaMA-2-7B: end-to-end Overall accuracy (%) under three realizations of π .

lengthen, holding above 82% Overall at $L \geq 5$ (full breakdown in Appendix C.3).

Model scale and storage complexity. We additionally evaluate CHRONOEXPERT and the controlled Shared-LoRA variant on the full TKE-BENCH-1K set using LLaMA-3.1-70B-Instruct and Qwen3-32B. Shared-LoRA trains one adapter for all time slices while holding the router and training objective fixed. Both methods retain the submitted LoRA configuration: rank 8, scaling factor 32, 50 epochs per slice, learning rate 10^{-3} , and KL weight 0.0625. We use edit layers 39 and 57, respectively. Table 6 reports the full-set results.

Within the LLaMA family, scaling from 8B to 70B improves Overall accuracy from 89.50 to 94.57. On the two larger backbones, per-slice isola-

Backbone	Method	Current	Before	After	Simple Entity	Simple Time	Overall
LLaMA-3.1-8B-Instruct	CHRONOEXPERT	89.60	82.85	82.46	90.83	97.27	89.50
LLaMA-3.1-70B-Instruct	CHRONOEXPERT	98.00	90.41	92.40	95.12	97.08	94.57
LLaMA-3.1-70B-Instruct	Shared-LoRA	36.45	38.29	31.71	53.86	80.75	52.75
Qwen3-32B	CHRONOEXPERT	80.65	75.76	76.14	84.62	84.91	81.19
Qwen3-32B	Shared-LoRA	25.75	34.66	23.91	34.93	72.18	42.20

Table 6: Performance on larger backbones. All values are accuracy percentages; the 8B result is included as a scale reference.

tion outperforms Shared-LoRA by 41.82 and 38.99 percentage points, respectively. The lower absolute result on Qwen3-32B means that these experiments support scalability and cross-family applicability, rather than a universal claim that accuracy increases monotonically with parameter count. Increasing backbone size alone does not eliminate temporal interference.

Each 8B time-slice expert contains 147,456 trainable parameters and occupies approximately 0.618 MB in FP32, including routing metadata. Storage therefore grows linearly as approximately $0.618N$ MB for N experts.

7 Conclusion

We introduced TKE-BENCH and used it to expose CTI: editing baselines collapse on chronological queries, which we trace to near-parallel same-chain subject-token keys. CHRONOEXPERT routes each query to its per-time-slice expert, lifting Overall accuracy above 85% on every backbone. Promising avenues include implicit-time queries and cross-relation chains.

Limitations

Our evaluation has several limitations. First, our answer-matching protocol (bidirectional substring for entity-valued questions and year-substring for simple_time) is a lightweight approximation to the answer-alias evaluation convention used by Yin et al. (2024), and occasionally over-credits aliases or partial matches; a stricter alias-aware subset would tighten the precision of the headline numbers. Second, our additional full-set experiments extend the evaluation to LLaMA-3.1-70B-Instruct and Qwen3-32B and show that per-time-slice isolation remains beneficial, but two additional backbones do not establish monotonic improvement with parameter count or generalization to all model families. Third, we edit and evaluate one temporal chain at a time and do not study the sequential,

lifelong setting in which many chains are edited together: there the router must disambiguate not only among the slices of a single chain but across the experts of all previously edited chains, and the expert store grows with the number of edited chains, so whether routing accuracy and storage stay manageable at that scale is left for future work.

Ethics Statement

Knowledge editing is dual-use: the same machinery that corrects outdated facts can also be misused to inject falsehoods. Two design properties make CHRONOEXPERT easier to audit than monolithic locate-and-edit methods. First, edits are confined to per-time-slice experts kept in a separate registry, so the base model is never permanently mutated and any chain can be rolled back by discarding its experts. Second, the router’s decision and the active expert at inference are explicit and inspectable, which supports downstream verification of which edited fact contributed to a generation.

TKE-BENCH is built entirely on the publicly available ATOKE-ME fact base, itself derived from YAGO 3.0.3; no personally identifying information is introduced beyond what is already present in YAGO. The paraphrased subtype is generated with GPT-4o under a gold-preserving prompt; the generation prompts and the manual spot-check protocol are documented in Appendix E.

Use of AI Assistants

We used general-purpose AI assistants only for language editing and minor productivity tasks, with all of their output reviewed by the authors. These tools did not contribute research ideas, experimental designs, scientific decisions, or any of the paper’s technical claims, results, or conclusions. All technical content, experimental results, and conclusions are the authors’ own, and the authors take full responsibility for the correctness of the paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2025YFB3003700) and the National Natural Science Foundation of China (Grant No. 62476025).

References

- AI@Meta. 2024. LLaMA 3 model card. <https://github.com/meta-llama/llama3>.
- Qizhou Chen, Dakan Wang, Taolin Zhang, Zaoming Yan, Chengsong You, Chengyu Wang, and Xiaofeng He. 2025. [Unedit: A unified knowledge editing benchmark for large language models](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc.
- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.
- Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6491–6506.
- Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.
- Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations*, volume 2025, pages 16366–16396.
- Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025. Lifelong knowledge editing requires better regularization. *arXiv preprint arXiv:2502.01636*.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.
- Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.
- Qwen Team. 2024. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Hua-jun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, and 1 others. 2024b. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024c. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.

Haoyu Xu, Pengxiang Lan, Enneng Yang, Guibing Guo, Jianzhe Zhao, Linying Jiang, and Xingwei Wang. 2025. Knowledge decoupling via orthogonal projection for lifelong editing of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13194–13213.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2024. History matters: Temporal knowledge editing in large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19413–19421.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

A Dataset Construction and Evaluation

A.1 Source Tuples and Chain Sampling

TKE-BENCH is built from the ATOKE-ME temporal fact backbone. Each source record is a time-scoped fact (s, r, o, t_s, t_u) with fields caseid, subject, relation, entity, since, and until. We group records by (s, r) , sort them by start year,

and treat each group as one temporal chain, then sample TKE-BENCH-1K by the relation-stratified procedure of §3.2 (chain length $2 \leq n \leq 7$, seed 42); the underlying chain and fact counts are reported there.

For each sampled chain, question generation is deterministic apart from the paraphrase step. Relation-specific templates first instantiate the five question types used in the paper: current, before, after, simple_entity, and simple_time. before questions are generated only for non-initial time slices, and after questions only for non-final time slices. We then use GPT-4o to generate one answer-preserving paraphrase per base question. The full paraphrase prompt appears in Appendix E.

A.2 Evaluated Question Counts

The raw construction pass emits a current question for each time-slice record, because the same question template is available on every record. In all reported experiments, however, current is evaluated once per chain and per subtype, against the final time slice. This avoids over-weighting the latest fact by the chain length and matches the semantics of “current”. The evaluated question counts are therefore exactly those in Table 7: 21,856 questions across the five question types and two subtypes.

qtype	orig.	para.	total
current	1,000	1,000	2,000
before	1,982	1,982	3,964
after	1,982	1,982	3,964
simple_entity	2,982	2,982	5,964
simple_time	2,982	2,982	5,964
total	10,928	10,928	21,856

Table 7: Question counts of TKE-BENCH-1K by question type and subtype. before/after are omitted at chain endpoints, and current is counted once per chain after deduplicating to the final slice.

A.3 Dataset Distributions

Table 8 reports the per-relation fact counts of TKE-BENCH-1K over its 2,982 temporal facts. The distribution is inherited from ATOKE-ME and is dominated by <playsFor>. Table 9 reports the chain-length distribution. In chain-length analyses (§6), chains of length $n \geq 5$ are merged into one bucket to keep the longest bucket large enough for stable comparison.

Relation	# facts	%
<playsFor>	2,654	89.00
<isMarriedTo>	178	5.97
<isAffiliatedTo>	87	2.92
<graduatedFrom>	33	1.11
<created>	12	0.40
<hasWonPrize>	6	0.20
<owns>	6	0.20
<worksAt>	4	0.13
<isLeaderOf>	2	0.07
total	2,982	100.00

Table 8: Relation distribution of TKE-BENCH-1K over the 2,982 temporal facts, inherited from ATOKe-ME.

n	2	3	4	5	6	7
# chains	502	229	130	82	38	19

Table 9: Chain-length distribution of TKE-BENCH-1K over 1,000 chains. Mean chain length is 2.98.

A.4 Answer Matching

We use the same matching protocol for all methods. For entity-valued question types, both the generated answer and the gold entity are normalized by lower-casing and stripping surrounding whitespace and punctuation; a prediction is correct if either normalized string is a substring of the other. Bidirectional substring matching is used because model generations often add a short explanation around the entity, while some gold strings contain qualifiers or aliases. For `simple_time`, we extract four-digit years from the generation and mark the answer correct if the gold start and end years are both present; for single-year intervals, the single gold year must be present. The same rule is used for original and paraphrased questions. This protocol follows the soft matching convention used in temporal knowledge editing, but we note in the limitations that it can occasionally over-credit partial aliases.

B Implementation Details

B.1 Baseline Implementations

All editing baselines are run on the same TKE-BENCH-1K split (§3.2), with each chain edited sequentially in chronological order and evaluated after all n edits have been applied.

ROME / MEMIT / ALPHAEDIT. We use the reference implementation from EasyEdit (Wang et al., 2024b). All three methods write updates to MLP down-projections at mid-layers, using the

subject-last token as the fact key. Shared hyperparameters: v -optimization runs for 25 gradient steps at learning rate 5×10^{-1} , the KL weight is 0.0625, and second-moment statistics are estimated from 100,000 Wikipedia samples. ROME writes a single layer ($\ell=5$ for GPT-J, LLaMA-2, and LLaMA-3.1). MEMIT distributes the update across five consecutive layers ($\ell \in [3, 7]$ for GPT-J, $\ell \in [4, 8]$ for LLaMA-2/3). ALPHAEDIT adds a null-space projection onto a preserved covariance with threshold 2×10^{-2} and L_2 weight 10; the projection matrix is precomputed once per backbone.

METO-augmented variants. METO adds a learned time embedding to the target v^* before the editor solves for ΔW . We follow the original time-conditional objective: the editor is given the calendar interval $[t_{s_i}, t_{u_i})$ alongside (s, r, o_i) and trained to predict o_i conditioned on the time tokens. All other hyperparameters match the underlying locate-and-edit method.

WISE. We use the reference implementation. The side memory is a copy of a single MLP down-projection: layer 20 `fc_out` for GPT-J, and layer 27 `down_proj` for LLaMA-2/3. Edit learning rate is 0.9, with 30 inner gradient steps, activation margins $[5.0, 10.0, 10.0]$, retrieval enabled, and 10 locality probes per chain drawn from a fixed locality pool. Edits within a chain are written into the side memory sequentially; at inference the method retrieves the best-matching memory slot for each query.

B.2 CHRONOEXPERT Expert Training

Each time-slice expert is a LoRA adapter on one MLP down-projection, with the base model frozen throughout. Table 10 lists the settings used in the main 1,000-chain experiments.

For a down-projection with input dimension d_{ff} and output dimension d , one expert stores $r(d_{\text{ff}} + d)$ trainable parameters. Expert storage therefore grows linearly in chain length, while the frozen backbone is shared across all chains and is never mutated.

B.3 Router Implementations

CHRONOEXPERT separates editing from routing. Given a chain, every candidate time slice is represented to the router as “entity, since to until”. We evaluate three concrete routers.

Heuristic router. The heuristic first looks for explicit years in the query and selects the inter-

Hyperparameter	Value
Injected module	MLP down_proj
LoRA rank r	8
LoRA scaling α	32
Optimizer	Adam
Learning rate	10^{-3}
KL weight γ	0.0625
Task weight λ	1
Epochs per slice	50
Base model	frozen
ℓ^* (GPT-J-6B)	21
ℓ^* (LLaMA-2-7B)	13
ℓ^* (LLaMA-3.1-8B-Instruct)	15

Table 10: Hyperparameters used by CHRONOEXPERT experts in the main 1,000-chain experiments.

val that contains the year. If no usable year is found, it checks whether an entity string from the chain appears in the query. As a final fallback, it routes current to the final slice. This router has no trainable parameters, but it struggles with before/after queries because the correct answer is a neighbor of the mentioned entity rather than the mentioned entity itself.

DeBERTa-MC router. The discriminative router is a DeBERTa-v3-base multiple-choice classifier. For a question with n candidate slices, we tokenize n pairs of the form question [SEP] candidate and select the candidate with the largest multiple-choice logit. The training set is built from the 7,819 ATOKE-ME chains outside TKE-BENCH-1K.

Qwen-Gen router. The default router is a Qwen2.5-0.5B generative scorer. It constructs a prompt listing the timeline candidates and the question, then scores each candidate string by the negative language-model loss of that candidate conditioned on the prompt. The selected slice is the candidate with maximum score. The router is fine-tuned for 3 epochs at learning rate 2×10^{-5} with batch size 16 on 24,207 question-level examples drawn from the same 7,819 ATOKE-ME chains outside TKE-BENCH-1K. The DeBERTa-MC variant uses the same supervision. The router prompt format is given in Appendix E.3.

Table 11 reports route-only accuracy, where the router prediction is evaluated directly against the gold slice without running the edited LLM. The Qwen-Gen router is nearly saturated on all question types, while DeBERTa-MC loses most of its errors on entity-anchored before/after queries.

Router	Cur	Bef	Aft	Ent	Tim	Overall
DeBERTa-MC	99.55	75.03	69.93	99.88	100.00	89.93
Qwen-Gen	100.00	99.62	99.72	99.83	100.00	99.84

Table 11: Route-only accuracy on TKE-BENCH-1K.

C Additional Experiments

C.1 Per-Time-Slice Isolation

Table 12 gives the per-question-type details for the Shared-LoRA ablation in Table 4. Shared-LoRA uses the same rank, scaling, edit layer, training objective, and Qwen-Gen router as CHRONOEXPERT, but trains one adapter for all time slices in a chain and keeps it active for all questions. It can still recover many simple_time questions because those queries directly mention the entity, but it collapses on state retrieval and entity-anchored navigation. This confirms that the main gain comes from separating time slices into disjoint parameter slots, not merely from using LoRA.

Backbone	Layer	Cur	Bef	Aft	Ent	Tim	Overall
GPT-J	21	27.55	3.81	5.71	25.86	76.43	32.15
LLaMA-2-7B	13	23.55	22.12	14.39	39.44	83.62	42.34
LLaMA-3.1-8B-Instruct	15	19.80	25.18	14.37	47.03	83.10	44.48

Table 12: Per-question-type accuracy of Shared-LoRA at each backbone’s deployed edit layer (GPT-J 21, LLaMA-2 13, LLaMA-3.1 15); CHRONOEXPERT is compared at the same layer.

C.2 End-to-End Router Ablation

Table 13 expands the router ablation on LLaMA-2-7B. All rows use the same per-time-slice LoRA experts at layer 13; only the router changes. The heuristic preserves current and simple_time, where explicit time or final-slice cues are available, but nearly fails on before/after. DeBERTa-MC recovers much of this gap, and Qwen-Gen is strongest on the anchor-based question types.

Router	Cur	Bef	Aft	Ent	Tim	Overall
Heuristic	87.15	0.83	0.33	41.77	94.58	45.37
DeBERTa-MC	87.80	59.54	56.17	87.37	94.72	78.70
Qwen-Gen	88.00	76.11	77.98	87.53	94.57	85.69

Table 13: End-to-end router ablation on LLaMA-2-7B.

C.3 Chain-Length Breakdown Across Backbones

Table 14 reports the chain-length breakdown across all three main backbones for CHRONOEXPERT and the strongest baselines. ROME and

ROME+METO are omitted, as they track the other locate-and-edit methods. The trend is consistent: locate-and-edit and memory-based baselines deteriorate as the number of time slices grows, while CHRONOEXPERT remains comparatively flat.

D Diagnosis Details

D.1 Temporal Key Collinearity Measurement

For the TKC analysis in §4.2, we run LLaMA-2-7B on all 1,000 chains of TKE-BENCH-1K without applying any edit. For each time slice, we use its time-prefixed prompt and locate the last token of the subject string. We then collect the hidden state at that token from layers $\{0, 3, 5, 7, 10, 13, 15, 17, 20, 25, 30\}$. For each layer, in-chain cosines are computed over all unordered pairs of time slices in the same chain, yielding 3,758 pairs. Cross-chain cosines are computed over 20,000 randomly sampled pairs whose case ids differ, using seed 0. All vectors are ℓ_2 -normalized before cosine computation.

Table 15 gives representative layers from the full curve in Fig. 1. In-chain subject-token states are nearly collinear at every layer. Cross-chain similarity is substantially lower at edit-relevant layers, so the near-collinearity is specific to time slices of the same subject rather than an artifact of hidden-state anisotropy alone.

E Question Generation Prompts

E.1 Manual Templates

Base questions for the five question types are instantiated by relation-specific templates. Table 16 shows representative templates for the dominant relation $r=\langle\text{playsFor}\rangle$; the other eight relations follow the same structural pattern with relation-appropriate verbs and object types (e.g., $\langle\text{isAffiliatedTo}\rangle$ replaces “team” with “organization” and “play for” with “be affiliated to”).

E.2 Paraphrase Prompt

The paraphrased subtype is generated by GPT-4o. The dataset construction code uses one batch prompt family for each question-type group: current, simple_entity, simple_time, and before/after. Figures 3–6 show only the paper-relevant paraphrased_question portion of each prompt template, matching the subtype evaluated in the main results. Any auxiliary fields emitted by the same batch scripts are dropped before constructing the reported benchmark split.

Outputs are parsed with a tolerant JSON parser, and records with missing or empty paraphrased_question fields are discarded. To assess paraphrase quality, we manually audit TKE-BENCH-1K with two graduate-student annotators. We draw a 10% sample stratified across question type and relation, so that low-frequency relations remain represented, and present each sampled item to both annotators. An item is judged *faithful* if its paraphrase is fluent and preserves the gold answer of the original question. The two annotators label every item independently and remain blind to each other’s judgments, and an item is counted faithful only when both accept it. Under this unanimous double-annotation criterion, more than 95% of the sampled items are judged faithful.

E.3 Router Prompt Format

The Qwen-Gen router scores candidate time slices under the prompt format in Figure 7. At inference we append each candidate string after Answer: and compute its negative language-model loss; the candidate with the largest score is selected.

Method	GPT-J				LLaMA-2-7B				LLaMA-3.1-8B-Instruct			
	$L=2$	$L=3$	$L=4$	$L \geq 5$	$L=2$	$L=3$	$L=4$	$L \geq 5$	$L=2$	$L=3$	$L=4$	$L \geq 5$
CHRONOEXPERT	85.54	90.14	88.19	86.57	87.86	87.24	84.93	82.26	91.78	89.72	87.44	87.96
WISE	4.16	1.78	1.15	0.99	24.19	14.74	11.06	9.73	26.25	19.43	14.44	10.97
MEMIT	38.58	26.05	19.43	13.84	34.60	21.03	14.97	10.78	35.60	23.75	17.53	13.74
ALPHAEDIT	38.68	25.91	18.38	13.75	40.56	27.84	22.25	15.28	29.30	21.88	14.45	11.80
MEMIT+METO	45.46	28.55	22.14	16.10	41.65	24.62	16.43	11.21	38.61	19.29	9.87	6.37
ALPHAEDIT+METO	39.65	26.55	20.02	14.63	48.34	32.34	26.35	18.44	35.51	25.32	17.97	13.72

Table 14: Overall accuracy by chain length across the three main backbones (each at its deployed edit layer), for CHRONOEXPERT and the strongest baselines; ROME variants follow the same trend and are omitted.

GPT-4o Prompt (current Paraphrase)

System Message:

You are an AI assistant that generates QA evaluation questions.

Instruction:

Given a fact triplet (subject, relation, object), a question asking for the object, the correct answer to the question should be the object in the triplet.

Task:

Create a paraphrased version of the original question. The correct answer should still be the object from the triplet.

Output Format:

Return ONLY a JSON object with key `paraphrased_question`.

Input Data:

- **Subject:** {s}
- **Relation:** {r}
- **Object:** {o}
- **Question:** {q}

Output:

Figure 3: Prompt used to generate the paraphrased variant for current questions. The template is instantiated from `dataset/prompt/current_generate.py`.

ℓ	In-chain		Cross-chain	
	mean	median	mean	median
5	0.9991	0.9995	0.1881	0.1667
13	0.9895	0.9939	0.5130	0.5021
20	0.9820	0.9890	0.6385	0.6974
25	0.9811	0.9893	0.6476	0.7019

Table 15: Representative TKC statistics on LLaMA-2-7B over all 1,000 chains of TKE-BENCH-1K.

qtype	Template ($r=\text{<playsFor>}$)
current	Which team does {s} currently play for?
simple_entity	Which team does {s} play for from { t_s } to { t_u }?
simple_time	When did {s} play for {o}?
before	Which team did {s} play for before {o}?
after	Which team did {s} play for after {o}?

Table 16: Manual question templates for the `<playsFor>` relation. Templates for the other eight relations follow the same pattern; the full set is generated by `final_question.py` in the released code.

GPT-4o Prompt (simple_entity Paraphrase)

System Message:

You are an AI assistant that generates QA evaluation questions.

Input Description:

- A fact triple (**subject**, **relation**, **object**).
- Explicit temporal boundaries **since** (**start-year**) and **until** (**end-year**).
- An original question whose correct answer is the **object** in the triple.

Task:

Rephrase the original question so that the expected answer is still the object.

Output Format:

Return ONLY a JSON object with the fixed key `paraphrased_question`.

Input Data:

- **Subject:** {s}
- **Relation:** {r}
- **Object:** {o}
- **since:** {t_s}
- **until:** {t_u}
- **original_question:** {q}

Output:

Figure 4: Prompt used to generate the paraphrased variant for `simple_entity` questions. The template is instantiated from `dataset/prompt/entity_generate.py`.

GPT-4o Prompt (simple_time Paraphrase)

System Message:

You are an AI assistant that generates QA evaluation questions.

Input Description:

- A fact triple (**subject**, **relation**, **object**).
- Explicit temporal boundaries **since** (**start-year**) and **until** (**end-year**).
- An original question asking when the object is valid for the subject and relation.

Task:

Rephrase the original question so that the answer is the same validity interval as in the original question.

Output Format:

Return ONLY a JSON object with key `paraphrased_question`.

Input Data:

- **Subject:** {s}
- **Relation:** {r}
- **Object:** {o}
- **since:** {t_s}
- **until:** {t_u}
- **original_question:** {q}

Output:

Figure 5: Prompt used to generate the paraphrased variant for `simple_time` questions. The evaluated answer remains the validity interval defined in §3.5; the template is instantiated from `dataset/prompt/time_generate.py`.

GPT-4o Prompt (before/after Paraphrase)

System Message:

You are an AI assistant that generates QA evaluation questions.

Input Description:

- A fact triple (**subject**, **relation**, **object**).
- Explicit temporal boundaries **since** (**start-year**) and **until** (**end-year**).
- An original question.

Task:

Rephrase the original question so that the answer is the same as the original question.

Output Format:

Return ONLY a JSON object with key `paraphrased_question`.

Input Data:

- **Subject:** {s}
- **Relation:** {r}
- **Object:** {o}
- **since:** {t_s}
- **until:** {t_u}
- **original_question:** {q}

Output:

Figure 6: Prompt used to generate the paraphrased variants for before and after questions. The original question determines whether the paraphrase asks for the predecessor or successor of the anchor entity; the template is instantiated from `dataset/prompt/before_generate.py`.

Qwen-Gen Router Prompt (Temporal Slice Selection)

Instruction:

You are a temporal knowledge router. Given a subject's timeline and a question, determine which time period the question refers to.

Input Data:

- **Timeline:** 1. {entity_1}, {since_1} to {until_1}; 2. {entity_2}, {since_2} to {until_2}; ...
- **Question:** {q}

Answer:

Figure 7: Prompt format used by the generative router.