TRAJEDIT: SAFETY EDITING WITH MULTI-ANCHOR TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Rapidly patching vulnerabilities in Large Language Models (LLMs) against evolving jailbreak attacks remains a critical challenge. While Knowledge Editing offers a resource-efficient alternative to retraining, existing methods apply supervision across broad portions of the target response, often constraining generic content irrelevant to the safe direction associated with the target correction. This inefficiency leads to severe over-refusal on benign queries, catastrophic forgetting of general capabilities, and limited generalization to broader safety-related tasks. In this paper, we propose TrajEdit, a solver-agnostic framework that reframes safety correction as sparse, multi-anchor trajectory editing. It constructs a safe trajectory for each selected anchor and jointly edits the corresponding divergence points, thereby reducing updates to generic content. We present an anchor selection strategy to identify anchor chunks associated with harmful generation. Extensive experiments across LLMs demonstrate that TrajEdit achieves superior defense success while maintaining a high non-refusal rate comparable to leading baselines. Additional evaluations show that the localized update better preserves general capabilities and transfers to other task settings, effectively reconciling the trade-off between safety and locality.

1 INTRODUCTION

Jailbreak attacks continually expose harmful behaviors in aligned large language models, while fully retraining or realigning a deployed model with RLHF (Ouyang et al., 2022) or DPO (Rafailov et al., 2023) is costly when vulnerabilities evolve rapidly. Because these failures concern behavior after alignment, post-hoc model editing can correct them without repeating the complete training process. Knowledge editing therefore provides a resource-efficient way to make targeted behavioral updates (Meng et al., 2022; 2023; Huang et al., 2026).

Recent safety editing methods, including DINM (Wang et al., 2024), UnKE (Deng et al., 2025), and AnyEdit (Jiang et al., 2025), typically apply dense token-level supervision across the target response. We refer to this full-response formulation as Global Editing, as illustrated in Figure 1(a). In an autoregressive response, harmful behavior may emerge at a specific transition. For example, a harmless preamble may turn into actionable harmful content, while later tokens only elaborate the behavior already selected. We use the term safety-critical turning point for such a transition. Global editing assigns supervision to all response tokens, mixing safety correction with full-sequence fitting and constraining content unrelated to the intended correction. As illustrated in Figure 1(a), this can reduce editing precision and damage benign capabilities.

These limitations expose an unresolved problem in safety editing: existing objectives do not distinguish safety-critical turning points from the surrounding generic content. This structure motivates the preliminary study in Section 2. Its PPL analysis in Appendix A identifies candidate locations, while the direction analysis tests whether the local update at those locations is aligned with the required correction. Because autoregressive generation unfolds step by step, we view the sequence of hidden states as a response trajectory and consider editing multiple locations along it.

To address this issue, we propose TrajEdit, a solver-agnostic framework for sparse, multi-anchor trajectory editing, as shown in Figure 1(b). TrajEdit selects anchor chunks associated with harmful generation and constructs a safe trajectory for each selected anchor, preserving the harmful prefix up to the point where the safe and harmful trajectories diverge. It then jointly edits the corresponding

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

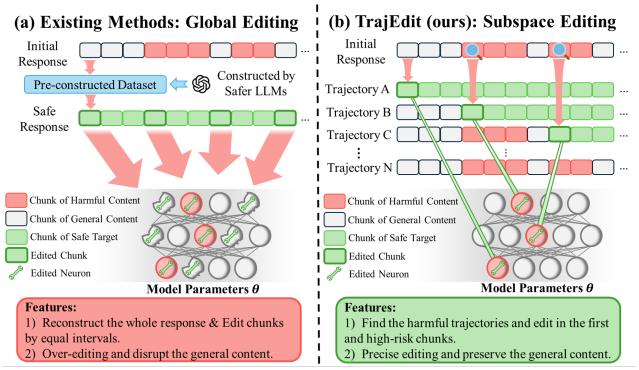


Figure 1: Comparison of TrajEdit and existing editing methods. (a) and (b) illustrates the process of existing methods and TrajEdit respectively.

Table 1: Similarity between anchor editing directions and the sample-level mean safe direction.

Model	First Chunk	PPL Location
Qwen3-8B	0.47	0.93
Llama-3.1-8B	0.71	0.91

divergence points, focusing the correction on safety-relevant locations while leaving generic content less constrained.

We evaluate TrajEdit across LLMs in sequential safety editing, general capability benchmarks, and cross-task settings. The results show that multi-anchor trajectory editing improves safety editing while reducing interference with capabilities unrelated to the targeted correction, and that the edited behavior can transfer beyond the original refusal setting.

Our contributions are summarized as follows:

- We propose TrajEdit, a solver-agnostic framework that constructs multiple safe trajectories from selected anchor chunks and jointly edits their corresponding divergence points.
- We present a chunk-level target construction strategy that produces a safe trajectory for each selected anchor chunk.
- Experiments across sequential editing, general capability benchmarks, and cross-task settings show stronger jailbreak repair, better preservation of benign capabilities and general utility, and transfer to other task settings.

2 Preliminary Study

Appendix A compares response-level and chunk-level PPL across the *unsafe head*, *unsafe segment*, and *safe segment* of harmful responses. The analysis places semantic transitions in high-PPL anchor chunks. This leaves open whether a high-PPL location is also suitable for a safety edit, rather than only a location with high uncertainty.

Thus, the PPL analysis identifies candidate locations, while the following direction analysis tests whether those locations are suitable for editing.

We test this question with hidden-state optimization within each sample. After splitting each harmful response into fixed-length chunks, we compare the first chunk with PPL-based locations. At each location, we initialize the target hidden state from the harmful trajectory and optimize it with gradient descent toward a safe continuation. The displacement defines the local editing direction. We compare the directions from the first chunk and PPL-based locations with the sample-level mean safe direction. Figure 2 shows the directions with t-SNE, and Table 1 gives their similarities in the original activation space.

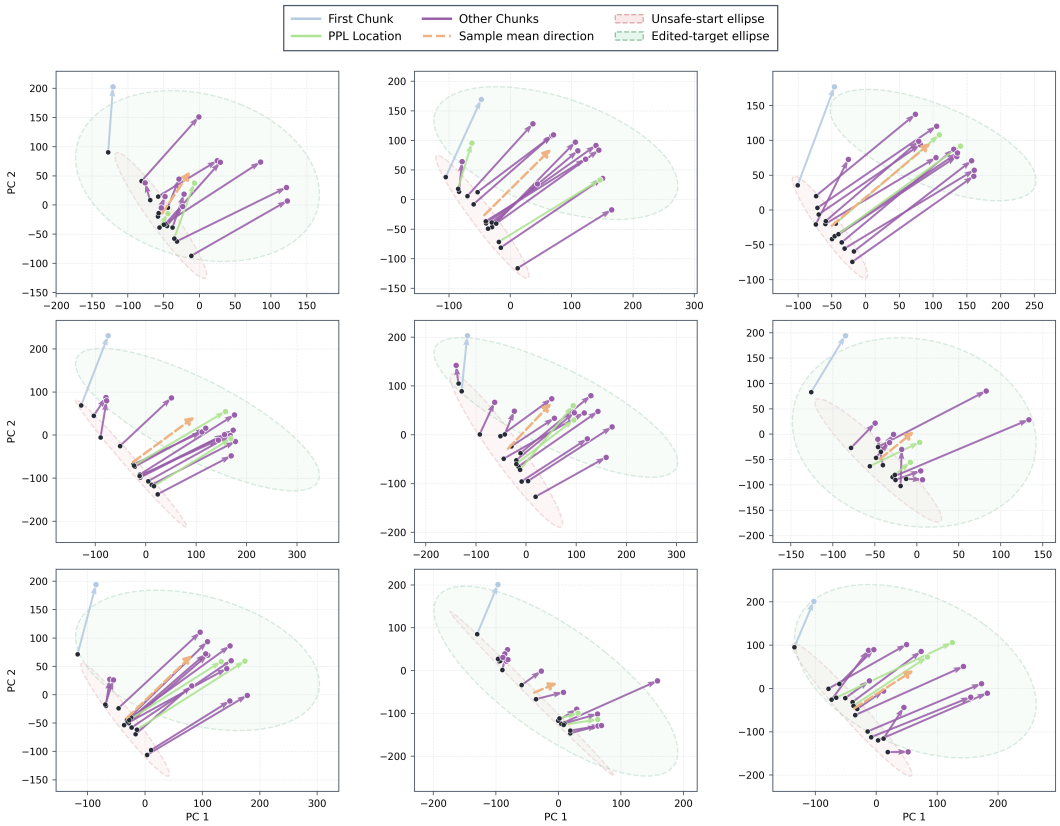


Figure 2: Anchor-level activation directions obtained by hidden-state optimization. The first chunk, PPL-selected anchors, and other chunks are compared with the sample-level mean safe direction across individual samples.

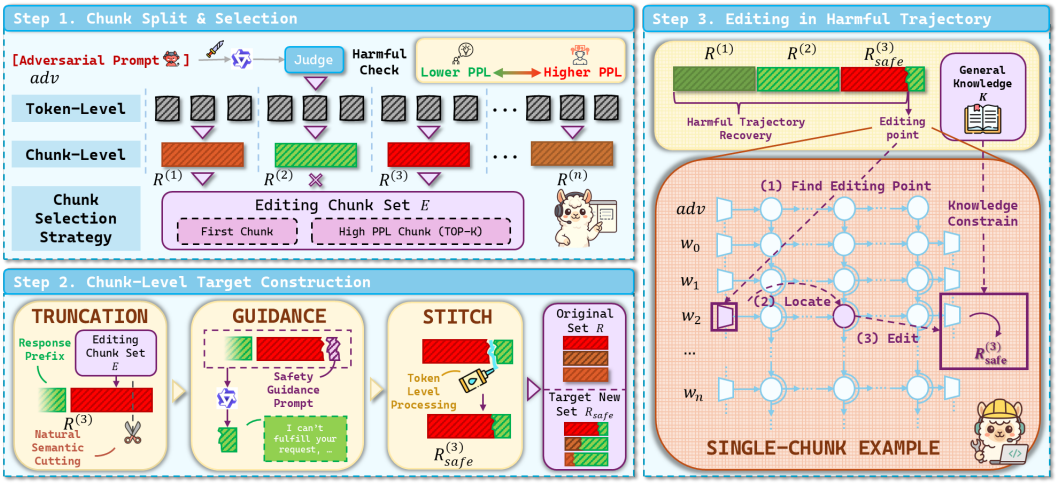


Figure 3: The overview of TrajEdit, consisting of three parts: **Chunk Split and Selection**, **Chunk-Level Target Construction**, and **Editing in Harmful Trajectory**.

Based on the results, we find that:

Across both models, the PPL-based directions are consistently closer to the mean safe direction than the first-chunk direction. This pattern shows that PPL localization identifies points where the required

hidden-state update is more aligned with the sample-level safety change, rather than only points associated with high uncertainty. The first chunk remains useful for constraining the entry of the harmful trajectory, while PPL-based anchors provide complementary locations for applying the safety edit. Together, they support the multi-anchor construction used by TrajEdit.

3 METHOD

3.1 OVERVIEW

Given an adversarial input A and its harmful response R_A, TrajEdit constructs multiple safe trajectories and uses their divergence points as editing constraints. As illustrated in Figure 3, the framework has three components: **Chunk Split and Selection**, which produces the editing chunk set E; **Chunk-Level Target Construction**, which constructs a safe trajectory for each anchor in E; and **Editing in Harmful Trajectory**, which applies the edit solver at the corresponding divergence points.

3.2 CHUNK SPLIT AND SELECTION

Given an adversarial input A that elicits an initial harmful response R_A, we split the response into fixed-length chunks:

$$R_A = [R^{(1)}; \ldots; R^{(n)}], \tag{1}$$

where each $R^{(i)}$ is a contiguous token subsequence. We denote the selected anchor indices by $E \subset \{1, \ldots, n\}$ and determine them using the following hybrid strategy.

We form E by combining the *first chunk* with the top-k chunks ranked by PPL-based uncertainty. The first chunk constrains the generation entry point, while the PPL-ranked chunks provide additional locations for correcting safety-relevant behavior along the response. This hybrid anchor set supplies the locations from which the safe trajectories in the next component are constructed.

The first chunk addresses behavior at the beginning of generation, where a harmful response can depart from the intended safety behavior. The additional uncertainty-ranked chunks cover later locations at which harmful content develops. Combining the two gives the editor access to both the response entry point and subsequent safety-relevant transitions.

3.3 CHUNK-LEVEL TARGET CONSTRUCTION FOR SAFE TRAJECTORIES

After identifying the anchor set E, we construct one safe trajectory for each selected location. For a given anchor chunk $R^{(i)}$, the target $R^{(i)}_{\text{safe}}$ shares the harmful prefix up to chunk i and then continues with a safe refusal. We obtain this target through the following steps:

The resulting pair preserves the original context before the anchor and changes the continuation after the selected location. Repeating this construction for every $i \in E$ yields paired trajectories with different divergence points, which provide multiple local constraints for the subsequent edit.

- **Step 1. Harmful trajectory recovery.** We restore the response prefix $[A; R^{(1)}; \ldots; R^{(i-1)}]$ at anchor i to recover the harmful generation trajectory.
- **Step 2. Natural semantic truncation.** Functional tokens, such as punctuation and articles, can reactivate contextual features during generation (Zhang et al., 2025). Within the anchor chunk, we randomly sample punctuation tokens as truncation points. These points retain the surrounding context while varying the editing location.
- **Step 3. Safe forward guidance.** At the truncation point, we insert a guidance prompt P that asks the model to generate a safe refusal from the harmful context. The resulting target uses the model's own continuation rather than a separately written response.
- **Step 4. Stitching the safe refusal.** We then remove P and concatenate the generated refusal with the prefix to obtain $R^{(i)}_{\text{safe}}$.

Iterating this for all anchors in E produces a set of alignment targets.

3.4 EDITING IN HARMFUL TRAJECTORY

Having obtained the paired trajectories for all $i \in E$, we use them to constrain the model at multiple anchor points. For each anchor, we define the local editing direction as the activation difference between the paired trajectories:

$$\Delta h^{(i)} = h^{(i)}_{\text{safe}} - h^{(i)}_{\text{harmful}}, \tag{2}$$

where the sample-level mean safe direction is obtained by averaging the local directions within the sample. The resulting editing objective is defined over the selected trajectories rather than over the entire response sequence.

3.4.1 OPTIMIZATION OBJECTIVE

The selected trajectories provide sparse constraints at the locations where the harmful and safe responses diverge. Rather than optimizing over the entire response, we restrict the objective to the selected anchors E; the relation to the full-sequence objective is derived in Appendix B:

$$\begin{aligned} h'^{(k)} =& h^{(k)} + \mathbb{I}_{k \in E} \cdot \arg\min_{\delta} \Big(\\ & - \log \mathbb{P}_{\theta(h^{(k)}+\delta)}[R^{(k)}_{\text{safe}} | A; R^{(1:k-1)}] \Big) \end{aligned} \tag{3}$$

By optimizing only the selected anchors in E, we focus the update on the paired trajectories and reduce unnecessary changes to generic content. This yields a sparse multi-trajectory alignment instead of a full-sequence fit.

3.4.2 SOLVER-AGNOSTIC WEIGHT UPDATE

The paired trajectories constructed in Section 3.3 define the constraints. The editing backbone then performs the corresponding neuron location and weight update. This formulation is solver-agnostic and can be integrated with locate-then-edit algorithms.

For each anchor pair, we identify the editing point at the last token of the common prefix, immediately before the first divergence between the harmful and safe trajectories. At this point, the chosen solver executes its standard two-phase update: locating critical neurons and editing their corresponding weights. Applying this procedure to all pairs in E minimizes the objective in Equation (3) over the selected trajectories.

The same procedure is applied to every paired trajectory in the anchor set. The resulting update therefore combines constraints from multiple points along the response while leaving the solver's parameter update rule unchanged.

3.5 DYNAMIC EDITING FRAMEWORK

Sequential editing across multiple samples can accumulate redundant updates, distort the model's representation space, and degrade general capabilities (Hu et al., 2024). To reduce this effect, AutoSafeEdit applies a classify-before-edit mechanism, as illustrated in Figure 4. At each editing step t, given an adversarial input A, we first evaluate the response generated by the current model θ_t. The parameter update is executed only when an external discriminator classifies the response as *Harmful*; otherwise, the update is skipped and θ_t is retained. AutoSafeEdit is applied uniformly to TrajEdit and the applicable baselines in sequential editing experiments.

4 EXPERIMENTS

In this section, we conduct experiments to answer the following research questions:

- **RQ1:** How does our method compare with existing approaches in the sequential editing setting?
- **RQ2:** How does the number of selected anchor chunks affect editing performance?
- **RQ3:** How well does editing preserve the model's general capabilities?
- **RQ4:** Does the edited model learn only to reject, or does it learn to defend across task scenarios?

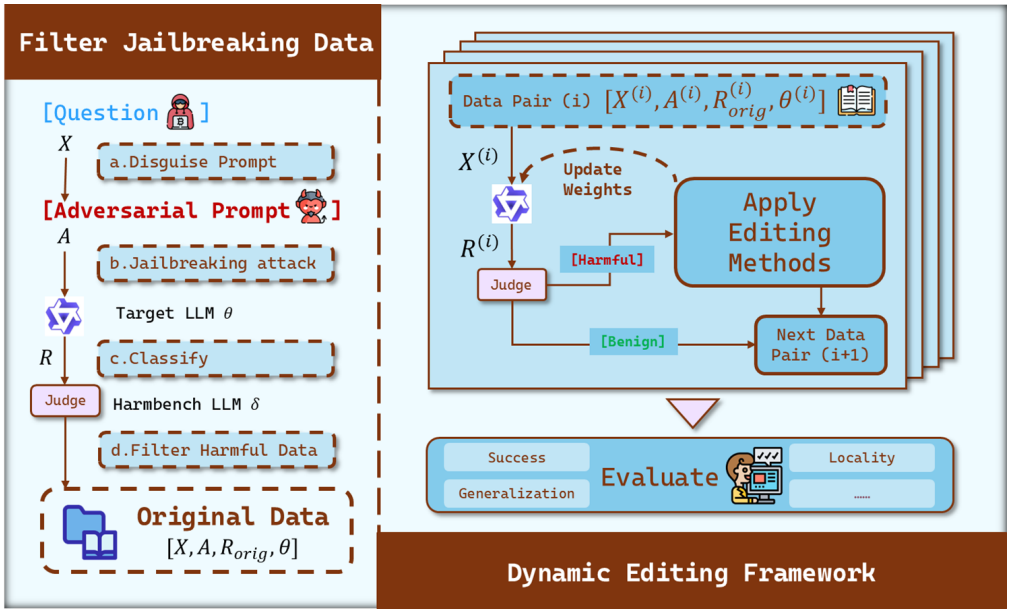


Figure 4: An overview of AutoSafeEdit, which applies a classify-before-edit mechanism to reduce redundant updates in sequential safety editing.

4.1 Experimental Setup

Datasets & Models. We evaluate on the SafeEdit benchmark (Wang et al., 2024), which contains 1,350 verified jailbreak instances across 9 safety categories and several attack patterns. We use the AlpacaEval set (Li et al., 2023) for locality and general capability assessment. Experiments are conducted on **Qwen3-8B** (Qwen Team, 2025) and **Llama-3.1-8B** (Llama Team, 2024). Dataset construction and sample selection details are provided in Appendix C.

Baselines. We compare TrajEdit against UnKE (Deng et al., 2025), DINM (Wang et al., 2024), AlphaEdit (Fang et al., 2025), and MEMIT (Meng et al., 2023). All baselines except DINM use the AnyEdit extension (Jiang et al., 2025) and the same dynamic editing framework described in Section 3.5.

Metrics. We report **Success**, **Generalization**, and **Locality**. Success measures refusal of the edited harmful query, Generalization measures transfer across questions and attacks, and Locality includes Fluency, ΔPPL, and **Non-refusal Rate**. Definitions and implementation details are provided in Appendix C.

4.2 Performance in Sequential Editing (RQ1)

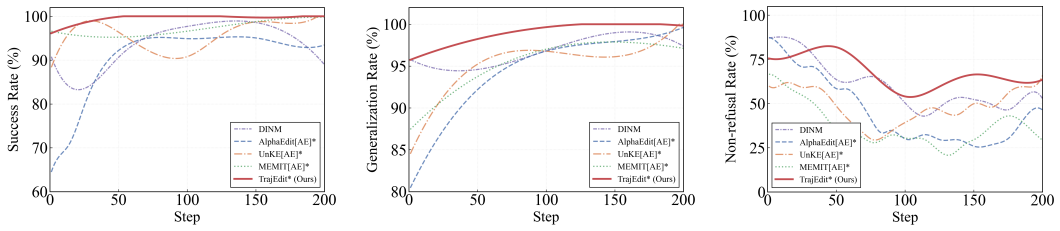
To evaluate the performance of various editing methods in the sequential editing setting, we apply both our proposed method and four baseline approaches on the safety dataset. The dynamic editing framework is employed across all methods to mitigate over-editing. Figure 5 illustrates the dynamic trends, while Table 2 details the aggregated metrics. Based on these results, we draw the following conclusions:

As detailed in Table 2, TrajEdit consistently outperforms baselines, achieving a defense success rate of **99.64%** on Qwen3-8B with improvements of 1.42% and 2.13% in success and generalization, respectively. It also surpasses the strongest baseline by over **21.89%** in Non-refusal Rate. As illustrated in Figure 5, TrajEdit reaches a stable safety performance within 50 editing rounds.

To validate the reliability of our editing method, we evaluated TrajEdit's editing performance on a broader range of standard harmfulness testing benchmarks—including HarmBench (Mazeika et al., 2024), JailbreakBench (Chao et al., 2024), and AdvBench (Zou et al., 2023). The experimental results

Table 2: Comparison of TrajEdit with existing methods applied on safety editing tasks on the sequential mode. $Succ.$, $Gen.$, $Loc.$, $Nrr.$ and $Flu.$ denote Success Rate, Generalization, Locality, Non-refusal Rate and Fluency respectively. Best results are highlighted in **bold**, while the second-best results are underlined. Methods using dynamic editing framework AutoSafeEdit are marked with '*'. Methods that are optimized with AnyEdit to fit long-text editing form are added a suffix of [AE]. Vanilla represents the performance before editing.

Model	Methods	Succ.↑	Gen.↑				Loc.			Time↓
			OnlyQ	OtherA	OtherQ	OtherAQ	ΔPPL↓	Nrr.↑	Flu.↑	
Qwen3-8B	Vanilla	33.81	91.10	44.66	60.68	71.35	-	74.71	8.35	-
	DINM	98.40	98.25	97.48	98.06	99.81	7.80	37.67	4.42	2.40
	MEMIT$_{[AE]}$*	93.78	97.69	69.22	91.99	85.05	**0.13**	69.54	7.97	5.26
	AlphaEdit$_{[AE]}$*	95.02	98.58	95.37	97.33	98.58	0.33	43.42	7.43	1.83
	UnKE$_{[AE]}$*	98.22	98.58	93.95	98.58	98.93	0.43	47.86	8.19	5.48
	TrajEdit* (ours)	**99.64**	**99.29**	**99.64**	**99.82**	**99.82**	0.17	**69.75**	**8.20**	**1.12**
Llama-3.1-8B	Vanilla	36.26	68.96	28.57	58.79	75.82	-	67.58	7.84	-
	DINM	98.44	96.44	99.22	**99.44**	99.67	11.91	57.11	3.62	1.46
	MEMIT$_{[AE]}$*	98.67	85.78	94.78	98.89	99.67	0.27	69.00	7.54	2.8
	AlphaEdit$_{[AE]}$*	98.78	**98.00**	97.56	98.89	99.44	**0.20**	67.00	7.24	0.60
	UnKE$_{[AE]}$*	98.22	97.63	97.63	98.22	99.41	1.26	63.31	**7.85**	4.16
	TrajEdit* (ours)	**99.78**	94.56	**99.56**	99.22	**99.89**	0.22	**69.52**	7.84	**0.44**



(a) Success rate (↑) (b) Generalization rate (↑) (c) Non-refusal rate (↑)

Figure 5: **Performance comparison on editing metrics.** We report the moving average of (a) success rate, (b) generalization rate, and (c) non-refusal rate over editing steps. TrajEdit (red solid line) performs better than the baselines across the three metrics.

are presented in the Appendix D.3. Additionally, using the MultiJail (Deng et al., 2024) benchmark dataset, we further validated the editing performance of TrajEdit in a multilingual environment, which is detailed in the Appendix D.4.

4.3 Impact of anchor chunk quantity (RQ2)

In this section, to further investigate the impact of different anchor chunk combinations on editing effectiveness, we conduct sequential editing using various selection strategies and monitor the edit success rate, generalization, and Non-refusal Rate under different anchor chunk quantities. We perform ablation studies on two key variables: the inclusion of the first text chunk and the number of top-k uncertainty anchor chunks selected. The results are presented in Table 3. We arrive at the following conclusion:

The combination of the first chunk and top-2 uncertainty anchors performs better on most evaluation metrics and provides a stronger overall trade-off than single-strategy baselines. This supports the use of multiple high-uncertainty anchors for more stable editing.

Table 3: Ablation experiments on chunk selection strategies. $Succ.$, $Gen.$, $Loc.$, $Nrr.$ and $Flu.$ denote Success Rate, Generalization, Locality, Non-refusal Rate and Fluency respectively. The best results are highlighted in **bold**.

ID	Settings		Metrics				
	First Chunk	Top-K	Suc. ↑	Gen. ↑	ΔPPL ↓	Nrr. ↑	Flu. ↑
0	×	0	55.50%	85.25%	-	72.00%	8.39
1	✓	0	99.00%	97.63%	0.33	**76.50%**	7.86
2	×	1	96.00%	96.38%	0.37	63.00%	8.08
3	✓	1	**99.50%**	**99.00%**	0.19	74.50%	7.54
4	✓	2	**99.50%**	**99.00%**	**0.18**	67.50%	**8.09**
5	✓	3	99.00%	98.50%	0.21	65.50%	8.01

Table 4: General capability benchmark results on Qwen3-8B before and after editing.

Method	MMLU	MATH	IfBench	LiveCodeBench
Vanilla (No-edit)	0.4125	0.1020	0.2600	0.3934
$\text{UnKE}_{[\text{AE}]}$	0.2315	0.0000	0.1633	0.0000
$\text{UnKE}_{[\text{TE}]}$ (ours)	0.3345	0.1050	0.2433	0.3763

4.4 General capability preservation (RQ3)

We further evaluate whether editing preserves the model's general capabilities. Using the Qwen3-8B model, we compare the unedited model (Vanilla), the baseline editing method UnKE with AnyEdit ($\text{UnKE}_{[\text{AE}]}$), and UnKE enhanced with TrajEdit ($\text{UnKE}_{[\text{TE}]}$). We report the performance on MMLU (Hendrycks et al., 2021a), MATH (Hendrycks et al., 2021b), IfBench (Pyatkin et al., 2025), and LiveCodeBench (Jain et al., 2025), which assess general knowledge, mathematical reasoning, instruction following, and code generation, respectively. The results are shown in Table 4. According to the results, we find that:

Compared with $\text{UnKE}_{[\text{AE}]}$, $\text{UnKE}_{[\text{TE}]}$ preserves substantially higher performance across all four benchmarks. The consistent improvements across general knowledge, mathematical reasoning, instruction following, and code generation indicate that multi-trajectory editing better preserves general capabilities during editing.

4.5 Cross-task editing transfer (RQ4)

To test whether the TrajEdit-edited model merely learned the rejection paradigm or achieved endogenous detoxification in terms of weights, we designed a set of different task scenarios to test whether the TrajEdit-edited behavior could be internalized and generalized to more real-world tasks. Starting from the agent design level, we set a risk identification task for the LLM, requiring it to determine whether the input is safe, and when identified as unsafe, to output in the format '<safety_warning>reason for unsafety</safety_warning>'. After a finite number of edits (n times) on the target model Qwen3-8B, we tested whether the model could identify risks while following instructions. The experimental results are shown in the Table 5. According to the results, we find that:

TrajEdit transfers the edited behavior from refusal to risk identification. TrajEdit achieves higher success and generalization than the compared editing methods in this task setting.

Additional experiments examine representation changes before and after editing in Appendix D.9, as well as the effects of editing layers, downstream solvers, chunk sizes, and the AutoSafeEdit framework in Appendices D.1, D.2, D.8, and D.10, respectively.

Table 5: Cross-task editing transfer results. n denotes the number of editing steps, and k denotes the number of selected anchor chunks. Methods using the dynamic editing framework AutoSafeEdit are marked with '*', while methods adapted with AnyEdit are marked with [AE].

Edit Steps	UnKE$_{[AE]}$	AlphaEdit$_{[AE]}$	MEMIT$_{[AE]}$	TrajEdit			TrajEdit*		
	/	/	/	1	2	3	1	2	3
n=0	48.89	48.89	48.89	48.89	48.89	48.89	48.89	48.89	48.89
n=5	0	57.04	50.81	72.30	88.22	83.04	57.56	98.74	85.19
n=10	0	50.81	53.63	100	90.32	97.26	91.85	100	23.75

5 Related work

5.1 Safety alignment challenges

Standard alignment techniques like RLHF (Ouyang et al., 2022) and DPO (Rafailov et al., 2023) are effective but computationally expensive, limiting their agility in patching emerging vulnerabilities. Adversaries exploit this rigidity via optimization-based attacks (e.g., GCG (Zou et al., 2023)) or semantic role-playing (e.g., AutoDAN (Liu et al., 2024)). While aggressive filtering can mitigate these threats, it often induces severe *over-refusal* (Röttger et al., 2024), where models decline harmless queries due to superficial keyword matching. This tension between robust defense and utility necessitates more surgical correction methods.

5.2 Unstructured knowledge editing

Knowledge Editing (KE) offers an alternative to retraining by modifying parameters associated with a target behavior. ROME (Meng et al., 2022) locates factual associations in the model and edits them through a localized parameter update, while MEMIT (Meng et al., 2023) extends this setting to mass-editing multiple associations. AlphaEdit (Fang et al., 2025) further constrains the update in a null space to preserve unrelated knowledge, and AnyEdit (Jiang et al., 2025) broadens the editing target beyond a fixed set of factual associations. Together, these studies establish the main editing objectives of changing target behavior while limiting interference with other model capabilities.

Safety editing applies these objectives to free-form responses and longer contexts. DINM (Wang et al., 2024) formulates detoxification as a knowledge editing problem, and UnKE (Deng et al., 2025) extends knowledge editing to unstructured data in large language models. AnyEdit (Jiang et al., 2025) also provides an editing framework for diverse knowledge types and is used to adapt existing editors to longer-text settings. Huang et al. study model editing as a double-edged sword for steering agent behavior toward beneficence or harm (Huang et al., 2026), emphasizing that behavioral correction must be evaluated together with its effect on general model behavior. These studies motivate evaluating safety editing with Success, Generalization, and Locality rather than with the target behavior alone.

6 Conclusions and future work

In this paper, we propose TrajEdit, a solver-agnostic framework for sparse, multi-anchor trajectory editing. It constructs a safe trajectory for each selected anchor chunk and jointly edits the corresponding divergence points, focusing the update on safety-relevant behavior. Across sequential safety editing, general capability benchmarks, and cross-task settings, TrajEdit improves Success and Generalization while maintaining Locality and Non-refusal Rate. The preliminary study shows that PPL locations provide editing directions that are more closely aligned with the sample-level mean safe direction, supporting their use together with the first chunk. The experiments on anchor selection, extreme response conditions, and chunk sizes further show that strong editing performance can be retained with a small number of selected chunks, yielding a favorable balance between editing effectiveness and editing time. Future work will investigate more precise anchor chunk selection and the extension of TrajEdit to other editing tasks involving unstructured knowledge. Its applicability to Multimodal Large Language Models and other task settings also remains to be explored.

AI USE STATEMENT

We used AI tools to assist with writing and revising code and with refactoring the project to improve code organization and maintainability. We also used these tools to polish the manuscript, including correcting grammar, refining sentence structure, and improving clarity and consistency of expression. The authors are responsible for the implementation, experimental results, and all claims presented in the paper, including content developed with AI assistance.

ETHICS STATEMENT

This research utilizes harmful prompts and jailbreak datasets strictly for defensive evaluation to enhance LLM safety alignment; all experiments were conducted in a controlled environment to prevent the dissemination of generated malicious content.

REFERENCES

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Datasets and Benchmarks Track*, volume 37, pp. 55005–55029, 2024. doi: 10.52202/079017-1745. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/63092d79154adebd7305dfd498cbff70-Abstract-Datasets_and_Benchmarks_Track.html.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. Everything is editable: Extend knowledge editing to unstructured data in large language models. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/02763667a5761ff92bb15d8751bcd223-Abstract-Conference.html.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. In *International Conference on Learning Representations 2024*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6b396f766a50e0853a5164e68048540c-Abstract-Conference.html.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/29c8c615b3187ee995029284702d3f43-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021) Datasets and Benchmarks Track*, volume 34, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. WilKE: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3476–3503, 2024. doi: 10.18653/v1/2024.findings-acl.207. URL https://aclanthology.org/2024.findings-acl.207/.

Baixiang Huang, Zhen Tan, Haoran Wang, Zijie Liu, Dawei Li, Ali Payani, Huan Liu, Tianlong Chen, and Kai Shu. Model editing as a double-edged sword: Steering agent behavior toward beneficence or harm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31113–31121, 2026. doi: 10.1609/aaai.v40i37.40372.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Anyedit: Edit any knowledge encoded in language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27510–27533. PMLR, 2025. URL https://proceedings.mlr.press/v267/jiang25b.html.

Solomon Kullback. *Information Theory and Statistics*. Courier Corporation, New York, second edition, 1997.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An Automatic Evaluator of Instruction-following Models, 2023. URL https://github.com/tatsu-lab/alpaca_eval.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=7Jwpw4qKkb.

Llama Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783v3.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 35181–35224. PMLR, 2024. URL https://proceedings.mlr.press/v235/mazeika24a.html.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022) Main Conference Track*, volume 35, pp. 17383–17393, 2022. doi: 10.52202/068431-1262. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27730–27744, 2022.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025) Datasets and Benchmarks Track*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/46499a0622ecf568b72d17b61e45dbd5-Abstract-Datasets_and_Benchmarks_Track.html.

Qwen Team. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, May 2025. URL https://arxiv.org/abs/2505.09388.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 53312–53325, 2023.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for*

Computational Linguistics (NAACL), pp. 5377–5400, 2024. doi: 10.18653/v1/2024.naacl-long.301. URL https://aclanthology.org/2024.naacl-long.301/.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://jmlr.org/papers/v9/vandermaaten08a.html.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. Detoxifying large language models via knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3093–3118, 2024. doi: 10.18653/v1/2024.acl-long.171. URL https://aclanthology.org/2024.acl-long.171/.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. AKEW: Assessing knowledge editing in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 15118–15133, 2024. doi: 10.18653/v1/2024.emnlp-main.843. URL https://aclanthology.org/2024.emnlp-main.843/.

Shaohua Zhang, Yuan Lin, and Hang Li. Memory retrieval and consolidation in large language models through function tokens, 2025. URL https://arxiv.org/abs/2510.08203.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 59420–59449, 2023.

A EXPERIMENT: PPL-BASED ANCHOR LOCALIZATION ANALYSIS

A.1 EXPERIMENTAL SETUP

We use the SafeEdit dataset (Wang et al., 2024) and its verified adversarial queries A, which are derived from harmful questions X and reliably elicit harmful content. The analysis uses 568 successfully jailbroken instances.

For chunk-level analysis, we randomly sampled 30 responses from the pool and split them into 418 chunks using a fixed chunk length of 20 tokens. Formally, given a chunk consisting of tokens $(t_1, \ldots, t_n)$ and its prefix context X', the perplexity (PPL) is calculated as:

$$\mathrm{PPL} = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log p\big(t_i \mid X'; t_{1:i-1}\big)\right) \tag{4}$$

where n is the length of the chunk (20 tokens in our experiments) and $p(\cdot)$ is the next-token distribution of the model conditioned on the prefix and previously generated tokens.

A.2 PPL-BASED ANCHOR LOCALIZATION

For Qwen3-8B, we split each harmful response R_A into fixed-length chunks to identify candidate anchors:

$$R_A = [R^{(1)}; R^{(2)}; \ldots; R^{(n)}], \tag{5}$$

where each chunk $R^{(i)}$ is a contiguous token subsequence. We use PPL as the uncertainty metric for the default anchor selection strategy.

We categorize the chunks by semantic content as *unsafe head* (the onset of harmful content), *unsafe segment* (the body of harmful content), and *safe segment*. Figure 6 compares uncertainty at the response and chunk levels.

Based on the results, we find that:

Chunk-level PPL reveals local changes in uncertainty associated with harmful content. The *unsafe head* exhibits higher uncertainty, and *unsafe segments* generally have higher PPL than *safe segments*. These local differences are less apparent at the response level, where safe and unsafe responses have overlapping PPL distributions. The chunk-level pattern supports using PPL to locate safety-relevant transitions and select candidate anchors within harmful responses.

B DETAILED DERIVATIONS

This section derives the trajectory-based editing objective from the information-theoretic formulation of unstructured editing and obtains the localized objective over selected anchors.

B.1 PAIRED ACTIVATION DIFFERENCES

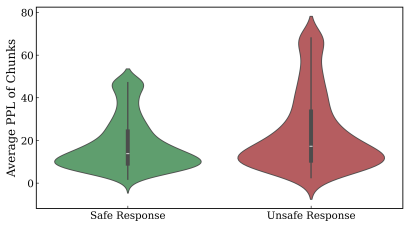
For each selected anchor, the derivation uses the local activation difference defined in Eq. (2). This sample-local quantity specifies the change required by the safe trajectory at the corresponding divergence point.

B.2 MUTUAL INFORMATION OBJECTIVE

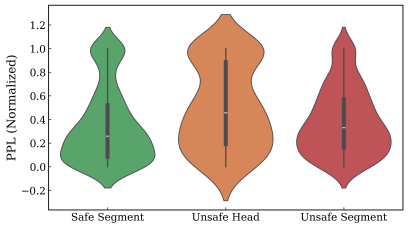
Let A be the adversarial input and R_{safe} be the target safe response. The goal of editing is to perturb the hidden states h' such that the mutual information between the input and the target response is maximized:

$$h' = \arg\max_{h'} I(A; R_{\text{safe}} \mid h'). \tag{6}$$

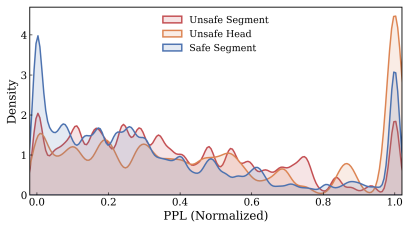
Here, $I(X; Y \mid Z)$ denotes the conditional mutual information (Kullback, 1997), which measures the reduction in uncertainty about X given Y when Z is known.



(a) Response-level PPL distribution



(b) Chunk-level PPL distribution



(c) Chunk-level PPL density estimation (KDE)



(d) Word cloud of high-PPL safe segments

Figure 6: **Analysis of perplexity (PPL) distributions.** (a) Response-level PPL comparison between safe and unsafe responses. (b) Chunk-level PPL distribution across safe segments, unsafe head, and unsafe segments. (c) KDE density plot comparing chunk-level PPL distributions. (d) Word cloud generated from the top 10% PPL of the safe segments.

When the response is split into multiple chunks $R = [R^{(1)}; R^{(2)}; \ldots; R^{(n)}]$, the global mutual information in Eq. (6) can be generalized to a multi-point perturbation problem. For a set of perturbed hidden states $h'^{(1:n)}$ across time steps, the objective becomes:

$$\max I\big(A; R_{\text{safe}} \mid h'^{(1:n)}\big). \tag{7}$$

Because the editing targets are specified at multiple paired anchors, we decompose Eq. (7) into a sum of local conditional mutual information terms:

$$\max \sum_{t=1}^{n} I\big(A; R_{\text{safe}}^{(t)} \mid R^{(1:t-1)}, h'^{(t)}\big). \tag{8}$$

For a specific chunk i, the local objective is to maximize the information provided by the perturbation $h'^{(i)}$ about the safe chunk $R_{\text{safe}}^{(i)}$, conditioned on the prefix:

$$\mathcal{L}^{(i)} = I\big(A; R_{\text{safe}}^{(i)} \mid R^{(1:i-1)}, h'^{(i)}\big). \tag{9}$$

B.3 Locality and Approximation

The paired trajectories indicate that the information needed for editing is concentrated near their divergence points.

- For **critical chunks** (e.g., refusal onset, harmful instructions), the paired trajectories provide a direct local editing constraint.
- For **neutral chunks** (e.g., conjunctions, generic explanations), editing provides little additional target-specific information.

Optimizing neutral chunks can therefore fit generic content and lead to overfitting. To distinguish informative anchors, we define the local safety information gain $\mathcal{I}^{(t)}$, which measures the contribution of the t-th chunk to the overall safety objective:

$$\mathcal{I}^{(t)} = I\big(A; R_{\text{safe}}^{(t)} \mid R^{(1:t-1)}, h'^{(t)}\big). \tag{10}$$

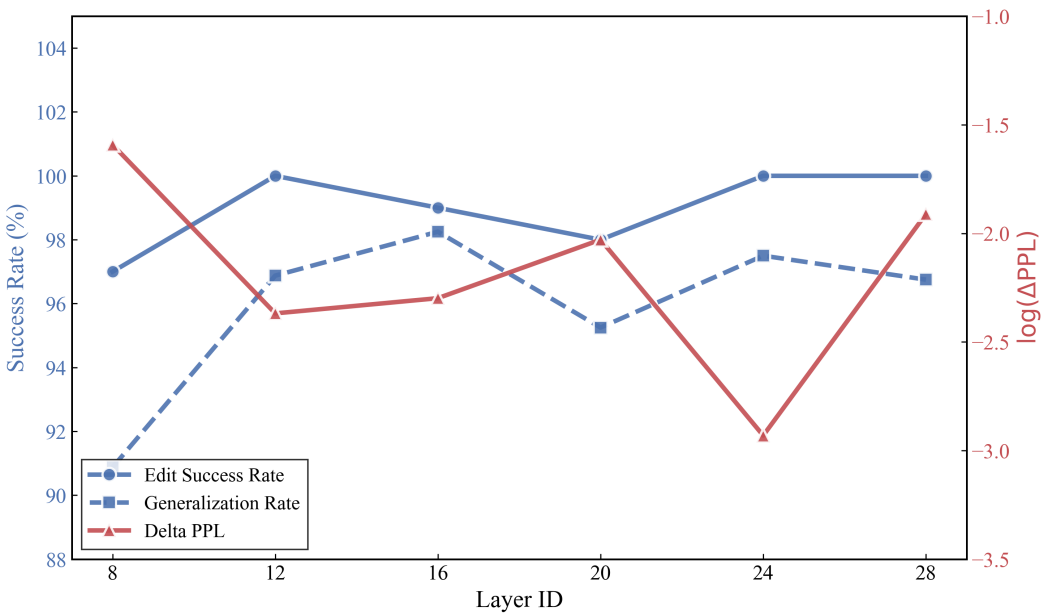
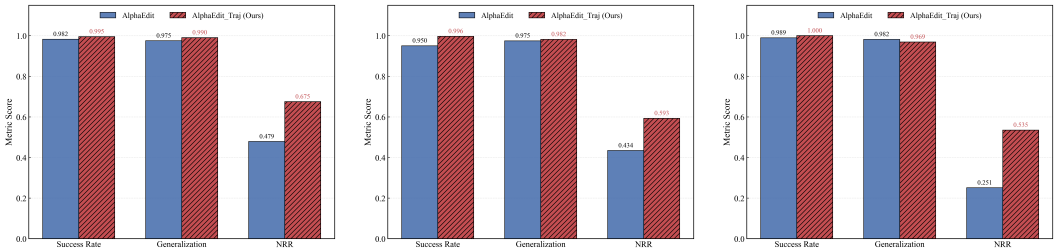


Figure 7: **Effect of edit layer selection.** Edit success rate, generalization rate, and $\log(\Delta\text{PPL})$ across selected layers.



(a) UnKE (b) AlphaEdit (c) MEMIT

Figure 8: **Method adaptation to different baselines.** Comparison of UnKE, AlphaEdit, and MEMIT before and after applying TrajEdit. TrajEdit improves success rate and Non-refusal Rate while maintaining strong generalization overall.

Consequently, the problem of finding the most effective editing locations reduces to a subset selection task. The optimal anchor set S is selected to maximize the cumulative information:

$$S = \arg \max_{S^*:\,|S^*|=k} \sum_{t \in S^*} \mathcal{I}^{(t)}. \tag{11}$$

This derivation motivates selecting a sparse anchor set. In our preliminary analysis (Appendix A), high-perplexity chunks provide a practical signal for selecting additional anchors. Thus, selecting the first chunk together with top-k uncertainty chunks gives an empirical approximation to the optimal set S, and the intractable global objective in Eq. (6) simplifies to the localized objective used in TrajEdit:

$$I\big(A; \{R^{(t)}_{\text{safe}}\}_{t \in S} \mid \{h'^{(t)}\}_{t \in S}\big). \tag{12}$$

This approximation focuses the editing budget on the selected trajectories and limits updates to unrelated knowledge.

Table 6: Editing performance of TrajEdit under multilingual settings. en, zh, it represent English, Chinese and Italian, respectively.

Language	Valid Samples	Success Rate
en	29	100%
zh	19	100%
it	64	100%

Table 7: Editing success rates on additional benchmarks. TrajEdit is compared with MEMIT, UnKE and AlphaEdit. Methods that are optimized with AnyEdit to fit long-text editing form are added a suffix of [AE].

Method	HarmBench	JailbreakBench	AdvBench
Vanilla	53.50	76.25	93.00
$\text{MEMIT}_{[\text{AE}]}$	73.75	92.00	98.33
$\text{UnKE}_{[\text{AE}]}$	<u>95.00</u>	<u>99.00</u>	**99.33**
$\text{AlphaEdit}_{[\text{AE}]}$	82.50	92.00	98.33
TrajEdit (ours)	**97.50**	**100.00**	<u>99.00</u>

C Implementation details

C.1 Datasets and evaluation metrics

Safety Dataset. The SafeEdit benchmark consists of 1,350 adversarial samples derived from 9 distinct categories (e.g., illegal activity, hate speech). These samples utilize diverse attack templates (e.g., GCG, AutoDAN) to rigorously test defense capabilities. For the Qwen3-8B and Llama-3.1-8B models, we selected 568 and 900 feasible jailbroken data through the discriminator respectively, as the datasets of the main experiment.

Irrelevant Knowledge. To verify locality, we use Wikipedia text and the wikiUpdate dataset from AKEW (Wu et al., 2024) for training (calculating covariance statistics), and the AlpacaEval set for evaluating the impact on general capabilities.

Metrics. The experiments use the following metrics:

- **Success**: The rate at which the edited model successfully refuses the specific harmful query it was trained on.
- **Generalization**: Includes four sub-metrics (OnlyQ, OtherA, OtherQ, OtherAQ) measuring defense transferability to unseen questions and attack templates.
- **Locality**: Measured by ΔPPL and generation entropy (Fluency) on irrelevant text. We also report the **Non-refusal Rate**, the percentage of benign AlpacaEval queries that the model answers without a safety refusal (Mazeika et al., 2024).
- **Time**: Average editing time per sample.

C.2 Discriminators and environment

Discriminators. To calculate the success rate and generalization, we employ HarmBench-Llama-2-13b-cls (Mazeika et al., 2024) as the jailbreak discriminator. For Non-refusal Rate, we use `Llama-3-8B` as the refusal discriminator following Chao et al. (2024).

Hardware. All experiments are conducted on a single NVIDIA A800 80GB GPU.

D Further experiments

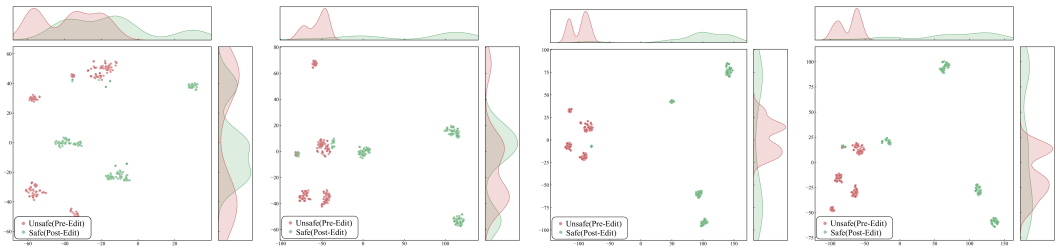
In this section, we evaluate **TrajEdit** under different editing settings and examine its editing effectiveness, generalization, and locality.

Table 8: Editing success rates in extreme response conditions.

Group	Average PPL	Average Fluency	Succ.(%)	Average Gen.(%)
Vanilla	-	-	35.00	68.25
Random Group	6.54	6.81	95.00	97.00
Low-PPL Group	3.05	6.90	98.00	98.25
High-Fluency Group	6.77	7.87	96.00	94.75

Table 9: Editing performance of TrajEdit under different chunk size.

Chunk Length (tokens)	0 (Before Editing)	20	40	60	80
Succ. (%)	36.00	99.50	**100.00**	**100.00**	**100.00**
Avg Gen. (%)	57.25	92.125	92.375	**95.25**	93.00
Nrr. (%)	67.00	**71.00**	68.50	68.00	67.50



(a) UnKE$_{[AE]}$ Safety (b) UnKE$_{[TE]}$ Safety (c) AlphaEdit$_{[AE]}$ Safety (d) AlphaEdit$_{[TE]}$ Safety

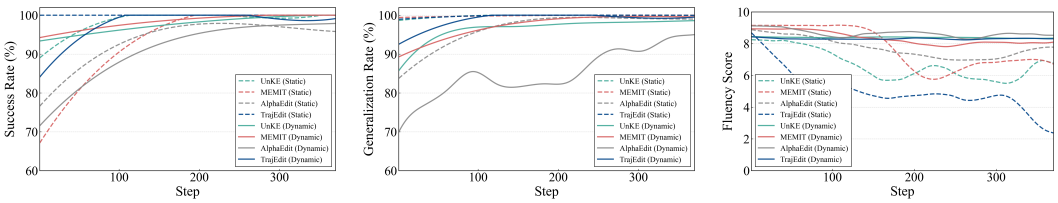
(e) UnKE$_{[AE]}$ Locality (f) UnKE$_{[TE]}$ Locality (g) AlphaEdit$_{[AE]}$ Locality (h) AlphaEdit$_{[TE]}$ Locality

Figure 9: **Visualization of hidden state distributions.** The **top row** shows the *safety* shift (red → green), where separation indicates the editing effect. The **bottom row** shows *knowledge locality* (blue → orange), where overlap indicates limited changes to other knowledge. Different columns compare the baselines applied with AnyEdit (marked with $_{[AE]}$) against TrajEdit (marked with $_{[TE]}$) across UnKE and AlphaEdit methods. TrajEdit separates the target hidden state distributions before and after editing while preserving substantial overlap for irrelevant knowledge.

D.1 Experiment: Impact of Editing Layer Selection

To examine the role of layer selection, we perform sequential editing on Qwen3-8B at different layers and evaluate editing effectiveness and locality. The results are shown in Figure 7. Based on the results, we find that:

TrajEdit maintains high editing success across the tested layers, providing flexibility in choosing where to apply the weight update. The middle and deeper layers generally combine strong generalization with better preservation of irrelevant knowledge. This pattern identifies a favorable range for layer selection and supports a trade-off between editing effectiveness and locality across multiple editing layers.



(a) Edit success rate (b) Edit generalization (c) Fluency score

Figure 10: A sequential editing comparison of dynamic editing framework used in four baselines: UnKE, MEMIT, AlphaEdit (all of them are applied with AnyEdit) and TrajEdit. Solid lines represent the *Dynamic* setting (using AutoSafeEdit framework), while dashed lines represent the default setting. **(a)** Success rate. **(b)** Generalization. **(c)** Fluency.

D.2 Experiment: adaptability to different solvers

In this section, we evaluate the adaptability of **TrajEdit** to different solvers using Qwen3-8B. We apply TrajEdit to **MEMIT**, **AlphaEdit**, and **UnKE**, and compare the edit success rate, generalization, and Non-refusal Rate before and after applying TrajEdit in sequential editing. The experimental results are shown in Figure 8. Based on the results, we find that:

TrajEdit improves editing success across the tested solvers while maintaining strong generalization overall. The accompanying increase in Non-refusal Rate shows that the edited models better preserve their ability to answer benign queries. Together, these improvements demonstrate a favorable trade-off between safety and locality across different editing algorithms, supporting the solver-agnostic formulation of multi-anchor trajectory editing.

D.3 Experiment: editing performance on more benchmarks

In this section, we evaluate the editing performance of TrajEdit in different benchmarks, including HarmBench (Mazeika et al., 2024), JailbreakBench (Chao et al., 2024) and AdvBench (Zou et al., 2023). These benchmarks focus on different categories of malicious requests and jailbreak attacks to test the security of general models and their attack defense capabilities in specific scenarios. The results are shown in Table 7. Based on the experimental results, we find that:

TrajEdit achieves the best or close to the best editing success across the evaluated benchmarks and improves defense success over Vanilla across datasets with different initial defense success rates. This consistent performance across diverse malicious requests and jailbreak attacks demonstrates the applicability of multi-anchor trajectory editing to a broad range of safety editing cases.

D.4 Experiment: editing in multilingual settings

In this section, we evaluate the editing performance of TrajEdit in multilingual settings. Based on the MultiJail dataset (Deng et al., 2024), we conduct experiments on the Qwen3-8B model in English, Chinese, and Italian. The results are shown in Table 6. Through the experiments, we find that:

TrajEdit successfully corrects the evaluated harmful responses in all tested languages. This supports the applicability of the editing procedure to harmful and safe trajectories in different languages, extending safety correction beyond English to the evaluated multilingual settings.

D.5 Experiment: comparison of chunk selection strategies

In this section, we evaluate the impact of different chunk selection strategies on editing effectiveness using Qwen3-8B and Llama-3.1-8B. The first chunk constrains the generation entry point, while PPL-ranked chunks provide additional locations for correcting safety-relevant behavior. The paired trajectories constructed at the selected anchors define the editing targets. We compare four strategies: **First Only**, which selects the first chunk; **Random**, which randomly selects three chunks; **First + PPL**, which combines the first chunk with the top-2 uncertainty anchor chunks; and **Full**, which selects all chunks.

Table 10: Comparison of chunk selection strategies over 50 non-sequential edits on each model. All results are percentages. Succ. denotes Success Rate, and Avg. Gen. denotes average Generalization over OnlyQ, OtherA, OtherQ, and OtherAQ. First + PPL selects the first chunk and the top-2 uncertainty anchor chunks, while Full selects all chunks. The best results for each model are highlighted in **bold**.

Model	Strategy	Succ.↑	Gen.↑				
			Avg. Gen.	OnlyQ	OtherA	OtherQ	OtherAQ
Qwen3-8B	First Only	82.00	57.50	88.00	**50.00**	54.00	38.00
	Random	34.00	49.00	88.00	40.00	30.00	38.00
	First + PPL	86.00	55.50	88.00	**50.00**	56.00	28.00
	Full	**90.00**	**60.00**	**90.00**	40.00	**58.00**	**52.00**
Llama-3.1-8B	First Only	90.00	59.00	**86.00**	38.00	72.00	40.00
	Random	40.00	35.50	62.00	24.00	34.00	22.00
	First + PPL	94.00	**66.00**	78.00	56.00	**82.00**	**48.00**
	Full	**96.00**	64.00	80.00	**58.00**	76.00	42.00

Table 11: Sample statistics and strict ROC-AUC for PPL-based localization. The percentages show the distribution of samples by the number of annotated `harmful_begin` chunks. Strict ROC-AUC evaluates chunk-level PPL against the exact annotated positions.

Model	Valid Samples	Avg. Chunks	`harmful_begin` Chunks (%)				Strict ROC-AUC↑
			1	2	3	≥ 4	
Qwen3-8B	435	29.97	78.85	16.78	3.22	1.15	0.8138
Llama-3.1-8B	521	29.99	79.46	9.40	5.57	5.57	0.8426

We conduct 50 non-sequential edits for each strategy on each model, applying each edit independently. Each harmful response contains 15 to 30 chunks, and First + PPL selects at most three anchor chunks. For each strategy, we construct the paired trajectories and edit the corresponding divergence points. Table 10 presents the edit success rate, the four generalization sub-metrics, and average generalization.

Based on the experimental results, we find that:

TrajEdit with First + PPL achieves performance close to the best results on most evaluation metrics by combining the first chunk with top-2 uncertainty anchors. The small number of selected chunks reduces editing time compared with Full while retaining comparable editing performance overall. This combination of effective safety correction and reduced editing time demonstrates the favorable trade-off between editing effectiveness and efficiency achieved by sparse multi-anchor trajectory editing.

D.6 Experiment: statistical validation of PPL-based localization

In this section, we evaluate PPL-based localization of safety-critical regions in harmful responses from Qwen3-8B and Llama-3.1-8B. We use Qwen3.5-9B to annotate chunks according to the onset of harmful content, actionable harmful guidance, and safety transitions, followed by manual review of the annotations. The label `harmful_begin` marks the onset of harmful content. Table 11 summarizes the number of valid samples, the average number of chunks, and the distribution of `harmful_begin` chunks per response.

We use chunk-level PPL to compute strict ROC-AUC, treating `harmful_begin` chunks as positives and all other chunks as negatives. We also evaluate chunk selection using Precision@k, Recall@k, F1, and Hit@k, allowing a distance of at most one chunk from an annotated `harmful_begin` position. We compare **First Only**, which selects the first chunk; **Top1 Only**, which selects the highest-PPL chunk; and **Top1 + First**, **Top2 + First**, and **Top3 + First**, which combine the first chunk with increasing numbers of top-ranked PPL chunks. Here, k denotes the total number of selected chunks, fixed for each strategy across samples with different numbers and positions of `harmful_begin` chunks. The chunk selection results are shown in Table 12.

Table 12: PPL-based localization under different chunk selection strategies. k denotes the total number of selected chunks. All metrics allow a distance of at most one chunk from an annotated `harmful_begin` position. The best results for each model are highlighted in **bold**.

Model	Strategy	k	Precision@k↑	Recall@k↑	F1↑	Hit@k↑
Qwen3-8B	First Only	1	**0.4989**	0.2061	0.2834	0.4989
	Top1 Only	1	0.3264	0.0947	0.1432	0.3264
	Top1 + First	2	0.4126	0.2884	0.3242	0.7149
	Top2 + First	3	0.3594	0.3547	0.3388	0.7770
	Top3 + First	4	0.3287	**0.4134**	**0.3469**	**0.8299**
Llama-3.1-8B	First Only	1	**0.8676**	0.3791	0.5146	0.8676
	Top1 Only	1	0.6334	0.2706	0.3683	0.6334
	Top1 + First	2	0.7505	0.5370	0.5944	0.8848
	Top2 + First	3	0.6436	0.6796	**0.6298**	0.9175
	Top3 + First	4	0.5170	**0.7314**	0.5745	**0.9328**

Based on the experimental results, we find that:

The strict ROC-AUC results show that chunk-level PPL distinguishes the onset of harmful content across both models. Combining the first chunk with PPL-ranked chunks consistently improves Recall@k and Hit@k, extending coverage of safety-critical regions along the response. This complementary pattern supports the hybrid anchor selection strategy: the first chunk constrains the generation entry point, while additional PPL-ranked chunks locate safety-relevant transitions within a sparse anchor set.

D.7 Experiment: anchor selection under extreme response conditions

We evaluate **TrajEdit** under extreme response conditions, including high-fluency and low-PPL cases. We select 100 samples for anchor-selection analysis and compare the Vanilla, Random, Low-PPL, and High-Fluency groups. We report average PPL and Fluency to characterize the response conditions, together with the edit success rate and average generalization. The results are shown in Table 8. Based on the results, we find that:

TrajEdit achieves strong editing success and generalization on low-PPL and high-fluency responses. The selected anchors enable effective safety correction when harmful content is generated with low uncertainty or high Fluency, and the edited behavior extends to other questions and attack templates. These results demonstrate effective anchor selection across the evaluated response conditions, with benefits for both editing success and generalization.

D.8 Experiment: impact of chunk size

In this section, we evaluate the impact of chunk size by varying the fixed chunk length from 20 to 80 tokens. We compare the edit success rate, average generalization, and Non-refusal Rate to examine how chunk length affects editing effectiveness and locality. The results are shown in Table 9. Based on the results, we observe that:

TrajEdit maintains high editing success and generalization across the tested chunk lengths, allowing flexibility in splitting harmful responses for anchor selection. The Non-refusal Rate also remains stable across chunk lengths and is preserved relative to the model before editing. These results demonstrate a consistent trade-off between safety and locality over the tested range, supporting the use of different chunk sizes in multi-anchor trajectory editing.

D.9 Experiment: representation space after few-shot sequential editing

To examine how multi-trajectory editing affects target behavior and irrelevant knowledge, we conduct a few-shot sequential editing experiment using 50 random instances followed by an evaluation on 200 held-out samples. We employ t-SNE (van der Maaten & Hinton, 2008) to visualize the hidden state distributions before and after editing for both target and irrelevant knowledge. We compare

UnKE and AlphaEdit with AnyEdit against their TrajEdit versions. As illustrated in Figure 9, we observe that:

The target hidden state distributions separate before and after applying TrajEdit, consistent with the change from harmful to safe behavior after few-shot sequential editing. At the same time, the hidden state distributions of irrelevant knowledge retain substantial overlap. These distinct changes in target and irrelevant knowledge support the locality of multi-trajectory editing, which concentrates the update on the target behavior while preserving irrelevant knowledge.

D.10 Experiment: dynamic editing framework analysis

We evaluate AutoSafeEdit in sequential editing by comparing the default setting with the dynamic editing framework for UnKE, MEMIT, AlphaEdit, and TrajEdit. We track the edit success rate, generalization, and Fluency as editing proceeds. The results are shown in Figure 10. Based on the results, we find that:

AutoSafeEdit maintains high editing success and generalization while preserving Fluency more effectively than the default setting as sequential editing proceeds. By applying updates to samples that elicit harmful responses, the dynamic editing framework reduces redundant updates and preserves Fluency over successive edits. These results demonstrate the benefit of AutoSafeEdit for maintaining both effective safety correction and locality during sequential editing.