The Illusion of Forgetting: Rank Leakage and Its Mitigation in Privacy-Preserving Knowledge Editing for Large Language Models

Jie Hou
George Mason University
jhou4@gmu.edu

Qiang Zeng
George Mason University
zeng@gmu.edu

Abstract

Privacy-preserving knowledge editing aims to protect sensitive information in large language models (LLMs) by rewriting their outputs. In this work, we identify a previously overlooked vulnerability of knowledge editing, which we term *rank leakage*: although editing replaces the top-1 output with a sanitized answer, the original sensitive token often remains highly ranked in the model's next-token distribution. Exploiting this phenomenon, we propose an effective black-box attack, called *RankPeek*, that exposes sensitive tokens by inspecting the top-ranked candidate tokens returned by a single model query. Despite its simplicity, RankPeek achieves strong extraction performance and remains effective even against models equipped with state-of-the-art defenses. To mitigate this vulnerability, we introduce **R**ank-guided **C**oncealment **E**diting (RCE), a plug-in defense that explicitly controls the rank of sensitive tokens during editing. Beyond RankPeek, RCE also mitigates existing black-box and white-box extraction attacks. Experiments across multiple models, datasets, and editing algorithms demonstrate that RCE substantially reduces extraction success while preserving edit fidelity and model utility. Our results highlight the importance of controlling the rank of sensitive tokens, rather than only the top-1 prediction, for privacy-preserving knowledge editing. Code and datasets are available at https://github.com/JasonH0810/RCE.

1 Introduction

Large language models (LLMs) have been shown to memorize sensitive training data, such as personal identifiers [6–8, 16, 22]. A growing body of work has demonstrated that memorized information can be extracted from language models through adversarial querying [3, 4, 16, 22, 25]. This raises serious privacy concerns, particularly under regulatory frameworks such as the General Data Protection Regulation (GDPR) [11], and underscores the need for methods that can remove sensitive knowledge from LLMs while preserving their utility.

A promising approach is *knowledge editing* (KE), which aims to rewrite specific factual associations in LLMs while preserving unrelated knowledge and model behavior [37, 40]. For example, KE can replace a memorized phone number or address with a non-sensitive one for privacy protection [27]. However, recent work has shown that this paradigm falls short of providing strong privacy guarantees [27]. In particular, even when edits succeed in changing the model's top-1 output, sensitive information can still be recovered via extraction attacks.

We identify a previously overlooked vulnerability in privacy-preserving KE, which we term *rank leakage*.[1] While prior work shows that sensitive information can remain in intermediate representations

[1] Here, "rank" refers to a token's position in descending order of output probability, rather than matrix rank in linear algebra. Higher-ranked tokens have smaller rank indices, with rank 1 denoting the most probable token.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

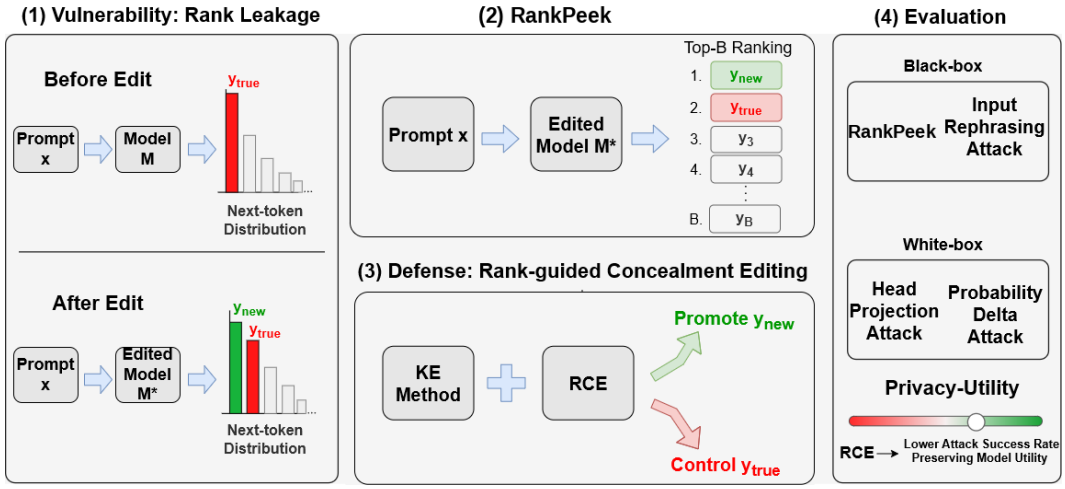

Figure 1: Overview of (1) rank leakage, (2) RankPeek attack, (3) RCE, and (4) the evaluation.

or be recovered through input rephrasing [27], we show that leakage already exists at the level of the final output distribution. Specifically, even after successful editing changes the top-1 prediction, the original sensitive token often remains highly ranked among candidate tokens. This residual signal is a readily exploitable channel for information leakage.

We thus introduce *RankPeek*, a simple yet effective black-box attack that exploits rank leakage by inspecting the top-ranked candidate tokens returned by a single model query. Under the same candidate-verification budgets used in prior work [27], RankPeek achieves high extraction success rates, even against models equipped with state-of-the-art defenses. Because RankPeek requires neither access to model internals nor repeated model queries, these findings demonstrate that rank leakage can be exploited under highly constrained adversarial capabilities.

To address this vulnerability, we propose **R**ank-guided **C**oncealment **E**diting (RCE), a defense that controls token ranks during editing. Figure 1 illustrates this pipeline, from rank leakage to attack and defense. Unlike probability-based erasure methods such as Fact Erasure [15], RCE promotes the intended new answer while pushing the original sensitive token beyond a chosen rank threshold, rather than simply minimizing its probability. This directly mitigates output-based extraction attacks, including the proposed *RankPeek* attack. Counterintuitively, our experiments show that overly large rank thresholds can increase the success rates of some extraction attacks and degrade model utility. This finding highlights the importance of calibrated rather than maximal rank suppression.

Although RCE applies its rank objective only to the final output distribution, logit-lens analysis shows that sensitive tokens also shift to lower-ranked positions in intermediate-layer projections. This observation helps explain RCE's improved robustness against representation-level extraction attacks. Moreover, RCE is method-agnostic and can be seamlessly integrated into existing KE frameworks as a lightweight *plug-in*, without modifying their core mechanisms.

We conduct comprehensive experiments across multiple editing methods, models, and datasets. While prior work [27] focuses on single-edit settings with single-token answers, we additionally evaluate RCE in more realistic *batch-edit* scenarios and on *multi-token* answers. Our results show that RCE substantially reduces extraction success rates across a wide range of settings, while maintaining high edit fidelity and competitive model utility in our primary experiments. For example, on CounterFact with AlphaEdit, RCE reduces RankPeek's attack success rate from 42.57% under HP Def to 11.87%, while achieving a Rewrite Score of 99.39%. Together, these findings show that privacy-preserving KE benefits from explicitly controlling the rank of sensitive tokens, rather than only the top-1 output.

Our contributions are summarized as follows:

- We identify *rank leakage* as a previously overlooked vulnerability in privacy-preserving KE, revealing that sensitive tokens often remain highly ranked after successful editing.

- We propose a black-box attack, called *RankPeek*, that exploits this vulnerability using only high-probability candidate tokens from a single model query, achieving strong extraction performance despite a highly constrained threat model.
- We introduce **RCE**, a plug-in defense that mitigates extraction attacks by pushing sensitive tokens beyond a chosen rank threshold while preserving the intended rewrite. We further reveal that overly large rank thresholds can increase extraction success and degrade model utility, highlighting the importance of carefully calibrated rank suppression.
- We demonstrate through extensive experiments that RCE significantly improves privacy protection across different editing methods and datasets, including batch-edit and multi-token settings, while maintaining high edit fidelity and competitive model utility in our primary evaluations.

2 Related Work

Privacy Leakage and Extraction Attacks in LLMs. Large language models (LLMs) can memorize and reproduce training data [4, 6]. This has motivated extensive study of privacy attacks. Membership inference determines whether a data point was used for training [5, 33], while attribute inference recovers latent properties of training data [20, 34]. Most relevant here are *data extraction attacks*, which aim to recover memorized content from language models [3, 4, 16, 22, 25]. Zhou et al. [41] use *repeated* generation to recover private documents or personal information from LLMs. Leak@k [30] examines whether sensitive information resurfaces across *repeated* probabilistic generations after unlearning. Our work studies a distinct failure mode of knowledge editing: even after a successful rewrite, the original sensitive token can remain directly exposed by a *single* query. We further develop RCE to control this residual rank exposure.

Auxiliary Guards, Machine Unlearning, and Knowledge Editing. External guard models such as Llama Guard [18] filter inputs or outputs at inference time, but the guard can be removed or modified when the model is deployed. To mitigate memorization within the model itself, prior work instead studies *machine unlearning* and *knowledge editing*. Machine unlearning removes the influence of specific data points [2, 38], but often incurs substantial degradation in model utility [39]. For example, REVS [1] applies discrete updates that can disrupt broader representations, leading to significant performance drops. Knowledge editing instead modifies specific factual associations while preserving general model behavior. Methods such as ROME [23] and MEMIT [24] localize edits to particular parameters, and AlphaEdit [12] further reduces interference.

Privacy-Preserving Knowledge Editing. Early work on privacy-preserving editing focuses on suppressing or rewriting sensitive outputs [9, 14, 26]. Recent work demonstrates that sensitive information can still persist in intermediate representations or be exposed under input rephrasing, and proposes defenses [27]. We identify an overlooked vulnerability, *rank leakage*, where sensitive tokens remain highly ranked in the output distribution after editing. We further propose a defense that explicitly controls sensitive-token ranks during editing.

3 Vulnerability and Attack

3.1 Rank Leakage: A New Output-Space Vulnerability

Knowledge editing modifies a model $\mathcal{M}$ to rewrite specific factual associations. After successful editing, the edited model $\mathcal{M}^*$ produces a sanitized top-1 prediction y_{new} instead of the original sensitive answer y_{true}. However, we find that y_{true} often remains highly ranked in the model's output distribution. We refer to this phenomenon as *rank leakage*.

To examine this phenomenon, we analyze the rank of the original sensitive token y_{true} after applying knowledge editing. We use the GPT-J-6B model [36] and the CounterFact dataset [23] in the single-edit setting (Section 5). Each instance is independently edited using AlphaEdit [12], MEMIT [24], or ROME [23], either with or without RCE.

Figure 2 shows the distribution of $\log_{10}(\text{rank}(y_{\text{true}}))$ across multiple editing methods. Across different knowledge editing methods, we observe a pattern: although the edited model successfully

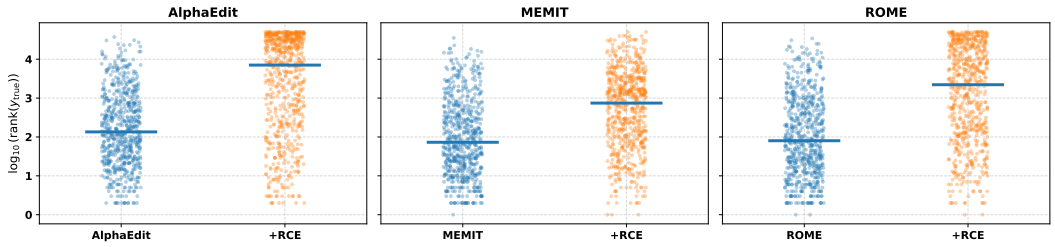

Figure 2: Distribution of $\log_{10}(\text{rank}(y_{\text{true}}))$ for different KE methods, with and without RCE. Results are obtained with GPT-J-6B [36] on CounterFact [23]. Across KE methods, the sensitive token often remains highly ranked after editing without RCE, whereas RCE shifts it toward lower-ranked positions (i.e., larger rank indices).

promotes y_{new} to the top-1 output, the original sensitive token y_{true} frequently remains within the top ranks of the vocabulary distribution. This behavior is consistent across KE methods, suggesting a general limitation of these KE approaches. This residual exposure motivates the RankPeek attack described next.

3.2 Threat Model

In practice, LLMs are typically deployed behind APIs, where users do not have access to model parameters or intermediate representations. Many model APIs, such as those provided by OpenAI and Google Gemini, return a small set of top-k candidate tokens (e.g., via `logprobs` parameters) [13].[2] In addition, real-world systems, including domain-specific applications such as medical or legal systems, may enforce rate limits and restrict repeated probing [17, 31, 32].

We thus consider a black-box adversary that makes a *single* query and observes only the top-k candidate tokens. (We additionally evaluate settings where only top-1 outputs are available; see Appendix B.8.)

Comparison with prior threat models. Existing extraction attacks typically assume stronger capabilities. White-box attacks, such as Head Projection (HP) and Probability Delta (PD), require access to model internals, while black-box attacks such as Input Rephrasing (IR) rely on multiple paraphrasing queries [27].

3.3 Exploiting Rank Leakage: RankPeek

We now introduce RankPeek (RP), which directly exploits rank leakage.

Attack Formulation. Following prior work [27], we assume a finite verification budget B, limiting the adversary to verifying at most B candidate answers to identify the true sensitive information (e.g., the attacker may only have enough budget to check B candidate addresses or call B candidate phone numbers). Given a prompt x and a verification budget $B \leq k$, the adversary queries the edited model $\mathcal{M}^*$ once and selects the B highest-probability tokens as the candidate set:

$$C(x) = \text{top-}B\ p(y \mid x; \mathcal{M}^*). \tag{1}$$

The attack succeeds if $y_{\text{true}} \in C(x)$.

Attack Success Metric. For an edited model $\mathcal{M}^*$ and an evaluation dataset of N target samples, we use **Attack-Success@B** to measure the fraction of samples for which the sensitive answer appears in the candidate set:

$$\text{Attack-Success@B}(\mathcal{M}^*) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y_{\text{true},i} \in C_i\},$$

[2] We conduct controlled experiments on locally hosted and edited models rather than commercial APIs because evaluating post-edit leakage requires knowing whether and what knowledge has been edited, information generally unavailable for commercial models.

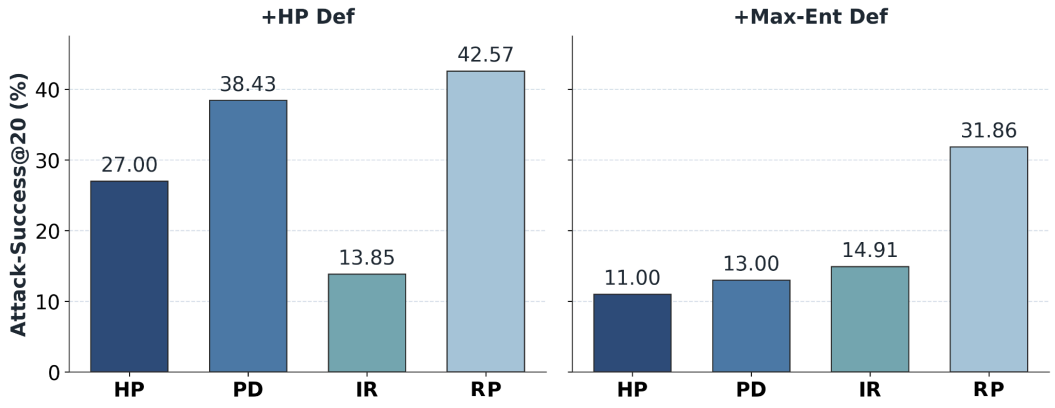

Figure 3: Attack success rates. Despite its simplicity, RankPeek (RP) consistently outperforms prior attacks (HP, PD, and IR [27]) against strong defenses (HP Def and Max-Ent Def [27]). Results are shown for AlphaEdit on CounterFact; additional results are provided in Section 6.

where C_i is the candidate set produced by the evaluated attack for the i-th sample, with $|C_i| \leq B$, and $\mathbf{1}\{\cdot\}$ is the indicator function. Higher values indicate stronger attacks.

Empirical Comparison with Prior Attacks. Figure 3 compares attack success rates under different defenses. At the same budget, RankPeek (RP) achieves higher success rates than prior attacks, HP, PD and IR, even against strong defenses such as Head Projection Defense (HP Def) and Max-Entropy Defense (Max-Ent Def) [27]. Unlike prior attacks, RP does not require input rephrasing, repeated queries, or access to internal representations.

Implications. The strong performance of RankPeek has two important implications. First, it demonstrates that rank leakage provides a powerful and previously underexplored attack surface. Even under a highly constrained threat model, sensitive information remains exposed among top-ranked candidates. Second, it reveals fundamental limitations of existing defenses. Current methods primarily focus on modifying the top-1 output or intermediate representations, while our findings highlight the need for new defense mechanisms that explicitly address rank leakage.

4 Mitigating Rank Leakage via Rank-Guided Concealment Editing

4.1 Overview

To mitigate rank leakage, we propose Rank-guided Concealment Editing (RCE), a plug-in method that augments existing knowledge editing approaches with rank-aware regularization. The key idea is to explicitly control the rank of the sensitive token y_{true} in the output distribution, rather than only changing the top-1 prediction. Specifically, RCE jointly promotes the intended new answer y_{new} to preserve edit fidelity and pushes y_{true} beyond a target rank threshold τ.

Importantly, RCE is designed as a *KE method-agnostic plug-in*: it can be seamlessly integrated into existing KE frameworks (e.g., ROME, MEMIT, AlphaEdit) without modifying their core pipelines.

4.2 Optimization Objective

Given an editing request $r = (x, y_{\text{true}}, y_{\text{new}})$, RCE computes a local update at a target layer L. In addition to the standard editing objective, which includes KL-divergence and weight-decay (WD) regularization [12, 23, 24], RCE introduces a rank-guided term to control the rank of y_{true}:

$$\mathcal{L} = \mathcal{L}_{new} + \lambda_{KL}\mathcal{L}_{KL} + \lambda_{WD}\mathcal{L}_{WD} + \lambda_{rank}\mathcal{L}_{rank}, \tag{2}$$

where $\mathcal{L}_{new}$ encourages the model to generate y_{new}, $\mathcal{L}_{KL}$ preserves the original output distribution for stability, and $\mathcal{L}_{WD}$ constrains the update magnitude to maintain locality.

Rank-guided penalty. To explicitly address residual leakage, we introduce a rank-based loss that encourages y_{true} to fall below a target rank boundary τ. Concretely, we define:

$$\mathcal{L}_{rank} = \max\big(0,\ \ell_{\text{true}} - \ell_{(\tau)} + m\big), \tag{3}$$

where ℓ_{true} is the logit of y_{true}, $\ell_{(\tau)}$ is the τ-th largest logit among all tokens excluding y_{true}, and $m > 0$ is a margin. This hinge-style surrogate avoids differentiating through discrete rank operators.

In contrast to probability-based penalties (such as those used in Fact Erasure [15]), this formulation explicitly targets the ranking structure of the output distribution, enabling fine-grained control over the rank of y_{true} and directly mitigating rank leakage.

5 Experimental Setup

Models. We conduct experiments with GPT-J-6B [36], GPT-2-XL [29], LLaMA-2-7B [35], LLaMA-3-8B [10], Qwen3.5-9B [28] and Ministral-3-8B [21]. We select these models because they are publicly available and widely used. Our main results are reported on GPT-J-6B, while results for the other models are provided in Appendix B.4.

Datasets. Following prior work [27], we use CounterFact [23] and zsRE [19] as our datasets. CounterFact is a widely used benchmark that constructs plausible counterfactual statements to deliberately conflict with facts memorized by the model, while zsRE consists of factual knowledge expressed in question–answer form. Following prior work [27], we retain only facts that the model knows before editing, since deleting knowledge is meaningful only when it is already encoded in the model. After filtering, CounterFact contains 657 instances and zsRE contains 586 instances. Details of the filtering procedure are provided in Appendix A.4. To investigate the applicability of RCE to realistic PII scenarios, we construct a controlled synthetic-PII memorization benchmark and evaluate RCE on it; the experimental setup and results are provided in Appendix B.11.

Knowledge Editing Methods. We employ three popular editing methods, AlphaEdit [12], MEMIT [24], and ROME [23]. The results on MEMIT [24] and ROME [23] are in Appendix B.3.

Defense Baselines. We compare RCE with several representative baselines, including strong defenses such as HP Def and Max-Ent Def [27] (see Appendix A.3 for details):

- **Head Projection Defense (HP Def) [27].** Reduces exposure of the original answer by enforcing margin-based constraints on logit-lens distributions across layers.
- **Max-Entropy Defense (Max-Ent Def) [27].** Encourages intermediate distributions to be more uniform, reducing the prominence of any individual token.
- **Empty Response [26].** Maximizes the probability of a fixed non-informative output (e.g., "dummy") instead of the original answer.
- **Fact Erasure [15].** Directly minimizes the probability of y_{true}.

We additionally report REVS [1] in Appendix B.5 as a recent privacy-preserving unlearning method. However, since REVS follows an unlearning paradigm rather than knowledge editing and does not enforce a specified target rewrite, we do not include it in the main comparison as a KE defense. We also omit Error Injection [9] from the main experiments, as prior work [27] shows that it provides very limited protection against extraction attacks.

Evaluation Metrics. In addition to **Attack-Success@B** (Section 3.3), we evaluate model utility and edit fidelity using the following metrics:

- **Random Δ-Accuracy [42].** To measure the impact on *model accuracy*, we compute $\Delta\text{-Acc}_{\text{rand}} = \text{Acc}(\mathcal{D}_{\text{rand}}, \mathcal{M}) - \text{Acc}(\mathcal{D}_{\text{rand}}, \mathcal{M}^*)$. For each dataset, we sample 100 unedited data points to form $\mathcal{D}_{\text{rand}}$ and use it across all editing instances and defense methods. Lower is better.
- **Neighborhood Δ-Accuracy [23].** Following prior work [27], we use the neighbor annotations provided in the dataset to construct a neighborhood set $\mathcal{D}_{\text{nbr}}$, consisting of prompts that

Table 1: Single-edit results on CounterFact and zsRE. Following prior work [27], we report Attack-Success@B with $B = 20$ for each attack method. $\Diamond$ and $\blacklozenge$ denote white-box and black-box attacks, respectively. Hit Rate and Rewrite Score are omitted for Empty Response and Fact Erasure, as these are *not* target-rewriting methods. Neighborhood Δ-Acc is *not* reported on zsRE because the dataset does not provide neighbor-prompt data.

Defense	Attack-Success@20 (%)↓				Δ-Acc (%)↓		Rewrite Score (%)↑	Hit Rate (%)↑
	HP$^{\Diamond}$	PD$^{\Diamond}$	IR$^{\blacklozenge}$	RP$^{\blacklozenge}$	Random	Neighborhood		
CounterFact								
AlphaEdit								
+HP Def	27.00	38.43	13.85	42.57	3.80	0.51	97.63	99.71
+Max-Ent Def	11.00	13.00	14.91	31.86	0.00	0.53	87.95	99.00
+Empty Response	55.25	63.31	19.63	31.35	0.05	0.13	–	–
+Fact Erasure	76.86	71.38	50.07	66.66	0.00	0.24	–	–
+RCE (ours)	11.71	17.47	3.65	11.87	1.43	0.85	99.39	99.71
zsRE								
AlphaEdit								
+HP Def	39.07	32.59	44.88	42.49	0.02	–	98.91	100.00
+Max-Ent Def	14.16	12.11	25.08	35.49	0.01	–	77.93	99.31
+Empty Response	61.20	57.85	45.40	50.10	0.02	–	–	–
+Fact Erasure	78.65	72.30	60.75	67.90	0.01	–	–	–
+RCE (ours)	30.88	25.76	28.83	34.64	0.01	–	90.06	99.31

share the same relation and ground-truth answer as the edited fact (e.g., *"Jack lives in NYC"* and *"Tom lives in NYC"*). We then quantify the change in accuracy on these neighboring examples before and after editing as $\Delta\text{-Acc}_{\text{nbr}} = \text{Acc}(\mathcal{D}_{\text{nbr}}, \mathcal{M}) - \text{Acc}(\mathcal{D}_{\text{nbr}}, \mathcal{M}^*)$. *Lower values indicate better locality preservation.* To further examine whether suppression effects spill beyond the neighborhood, we additionally construct semantically unrelated prompts that are independent of the edited subject and relation but share the same ground-truth answer. The corresponding setup and results are provided in Appendix B.9.

- **Rewrite Score [15].** Measures edit fidelity as the normalized increase in the target probability:

$$\text{RS} = \frac{p(y_{\text{new}} \mid x; \mathcal{M}^*) - p(y_{\text{new}} \mid x; \mathcal{M})}{1 - p(y_{\text{new}} \mid x; \mathcal{M})}. \tag{4}$$

  A value of 1 means that the edit perfectly maximizes the new target probability, while a value of 0 means that the new target probability did not change at all.

- **Hit Rate.** The fraction of prompts where the model outputs exactly y_{new}. This is reported only for methods that explicitly rewrite outputs.

Implementation Details. We follow the *single-edit* evaluation protocol used in prior work [27]. For each example, we apply a single edit to the model and evaluate performance before and after editing. We then restore the model parameters to their original state before processing the next example. All edits are evaluated independently, and results are averaged across examples. We additionally conduct *batch-edit* experiments. For all methods, we adopt the original hyperparameters reported in their respective papers to ensure fair comparison. Full implementation details and parameter studies are provided in Appendix A and Appendix B.2, respectively.

6 Experiment Results

6.1 What Is the Effectiveness of RCE Under Single-Edit Settings?

Performance of RankPeek. As shown in Table 1, despite being a black-box attack that requires only a single query, RankPeek (RP) frequently outperforms other attack methods (HP, PD, and IR [27]), highlighting the strength of this simple yet effective attack. For example, on CounterFact, even under a strong defense (Max-Ent Def), the success rate of RP reaches 31.86%.

Defense Effectiveness. As shown in Table 1, RCE exhibits strong defensive performance across attacks. In particular, RankPeek, which remains effective against other defense baselines, is significantly weakened by RCE, consistent with the rank-distribution shift observed in Fig. 2. It is

Table 2: Batch-edit results on CounterFact and zsRE. Unlike prior work [27], which evaluates only single edits, this setting applies hundreds of edits simultaneously. RCE achieves the lowest attack success rates across all four attacks on both datasets.

Defense	Attack-Success@20 (%)↓				Δ-Acc (%)↓		Rewrite Score (%)↑	Hit Rate (%)↑
	HP$^{\diamond}$	PD$^{\diamond}$	IR$^{\blacklozenge}$	RP$^{\blacklozenge}$	Random	Neighborhood		
CounterFact								
AlphaEdit								
+HP Def	24.20	31.05	13.84	33.63	22.40	5.17	96.90	99.39
+Max-Ent Def	21.00	33.18	17.65	29.37	27.90	13.68	24.72	71.99
+RCE (ours)	7.00	14.45	1.90	7.15	27.98	3.41	99.28	99.84
zsRE								
AlphaEdit								
+HP Def	37.37	47.09	16.89	51.70	7.86	–	91.47	97.78
+Max-Ent Def	43.00	49.31	27.64	52.90	8.64	–	15.55	32.08
+RCE (ours)	27.13	33.95	13.31	29.52	8.01	–	81.54	93.85

worth noting that, while both Max-Ent Def and RCE exhibit similar resilience to white-box attacks, Max-Ent Def is substantially weaker than RCE against the black-box RP attack on CounterFact (31.86% vs. 11.87%).

Edit Fidelity and Model Utility. The improved privacy robustness achieved by RCE does not come at the cost of edit fidelity. Specifically, Rewrite Score and Hit Rate both remain high, indicating that RCE reliably preserves the intended target outputs. Moreover, RCE maintains competitive Δ-Accuracy on both random and neighboring prompts, suggesting that the improved robustness is not achieved by degrading overall model utility. These results are also stable across repeated runs and evaluation-set variations; see Appendix A.5.

6.2 How Does RCE Behave Under Batch-Edit Settings?

Design. While prior work [27] only examined the single-edit setting, we also evaluate performance under batch editing, where models incorporate hundreds of edits simultaneously. In this setting, we evaluate only defenses that remain competitive in the single-edit setting, excluding methods such as Empty Response and Fact Erasure, whose defense effectiveness is poor in the single-edit results.

Defense Effectiveness. As shown in Table 2, RCE substantially reduces attack success rates on both datasets, outperforming other defense baselines. For example, on CounterFact, the success rate of RankPeek drops from 33.63% under HP Def to 7.15% under RCE.

Edit Fidelity and Model Utility. Importantly, the defense effectiveness of RCE does not come at the expense of editing fidelity or model utility. For example, on CounterFact, RCE attains a Rewrite Score of 99.28% and a Hit Rate of 99.84%, demonstrating that the intended target outputs are generated reliably in the batch-edit setting.

6.3 Why Does Output-Level Optimization Reduce Intermediate-Layer Exposure?

Although RCE applies its rank objective only to the final next-token distribution, we observe that it also reduces sensitive-token exposure in intermediate representations. Optimizing the output-level objective modifies the representation at the intervention layer to reduce the relative prominence of y_{true} in the final logits. Subsequent layers process this modified representation, so the effect of the update can propagate beyond the intervention layer. This provides a plausible explanation for why RCE can reduce later layers' tendency to re-amplify y_{true}, even without explicitly constraining their token ranks.

Consistent with this explanation, logit-lens analysis shows that y_{true} shifts to lower-ranked positions in intermediate states as well as in the final output distribution. This reduced exposure helps explain RCE's effectiveness against representation-level extraction attacks. Detailed experiments and logit-lens analyses are provided in Appendix B.7.

6.4 How Does First-Token Optimization Affect Multi-Token Sensitive Entities?

RCE optimizes only the rank of the first target token, while many sensitive entities are expressed as multi-token strings, including person names, locations, and short identifying phrases. This motivates the following question: to what extent does first-token suppression protect the corresponding multi-token string in practice? This aspect has not been studied in prior work [27].

We evaluate 299 multi-token targets using three complementary metrics: (1) **Greedy full recovery conditioned on first-token hit (GFR@FTH)** measures exact greedy recovery of the remaining target tokens after a first-token candidate-set hit. (2) **Stepwise recovery under the true prefix (StepR@TP)** measures whether each subsequent target token appears in the candidate set given the ground-truth prefix, conditioned on a first-token hit. (3) **Suffix recovery after first-token miss (SufR@FTM)** measures exact suffix recovery when greedy generation produces an incorrect first token.

Empirically, the results support this interpretation: suppressing the first token is associated with reduced leakage beyond the first position. Compared with AlphaEdit, RCE reduces GFR@FTH from 37.88% to 13.58% and StepR@TP from 79.55% to 30.86%. Moreover, once the first token is generated incorrectly, exact suffix recovery remains rare, suggesting that the first token often acts as a bottleneck for recovering the full sensitive string. Detailed metric definitions, protocols, and complete results are provided in Appendix B.10.

6.5 Adaptive Attack

Our defense aims to demote the sensitive token y_{true} beyond a rank threshold τ in the next-token distribution. This naturally raises an adaptive concern: rather than eliminating rank leakage, RCE might simply relocate y_{true} from the top-ranked region to a predictable neighborhood around the enforced threshold. An adaptive adversary aware of this objective may therefore concentrate its candidate-verification budget on a local rank window around τ, rather than the top-ranked tokens.

Specifically, given a prompt x and a candidate budget $B = 2w$, the attacker collects tokens whose ranks lie within $[\tau - w,\ \tau + w)$ and succeeds if y_{true} is included. Unlike our primary top-k black-box setting, this adaptive evaluation assumes access to token rankings around the defense threshold, representing a stronger adversarial capability. Although τ acts as a rank boundary rather than an exact target rank—the RCE objective only requires y_{true} to be pushed beyond this boundary—the optimization could still empirically cause sensitive tokens to concentrate near τ. This localized rank search therefore directly tests whether RCE creates such a predictable leakage region.

Table 3: Adaptive rank-local search attack success rates around $\tau = 5{,}000$ under varying budgets B. Even with larger budgets, success remains very low ($< 2\%$).

Window size B	20	50	100	200	400	800
Attack-Success@B (%)	0.00	0.00	0.15	0.15	0.76	1.98

As shown in Table 3, searching around the rank threshold τ yields low extraction success rates. Even when the candidate budget increases to 800, the attack success rate is only 1.98%. This result is consistent with Figure 2, which shows that $\mathrm{rank}(y_{\text{true}})$ does not concentrate around $\tau = 5{,}000$. These results suggest that RCE does not simply concentrate sensitive tokens in a narrow neighborhood around the rank threshold. Detailed experiments and analyses are provided in Appendix B.1.

6.6 Parameter Study

We study the two key hyperparameters of RCE, the rank threshold τ and the margin m, under batch editing, with full results provided in Appendix B.2. We focus here on τ, which specifies the rank boundary beyond which the sensitive token y_{true} is encouraged to move. At first glance, one might expect that setting τ as large as possible—e.g., close to the vocabulary size—would always provide stronger protection. Indeed, as τ increases, the τ-th ranked logit becomes smaller, making the rank constraint more stringent and requiring y_{true} to be pushed below an increasingly deeper rank boundary.

Our results show that this intuition holds only up to a point. As shown in Figure 4, increasing τ from small values initially improves resistance to extraction attacks, but the benefit quickly saturates

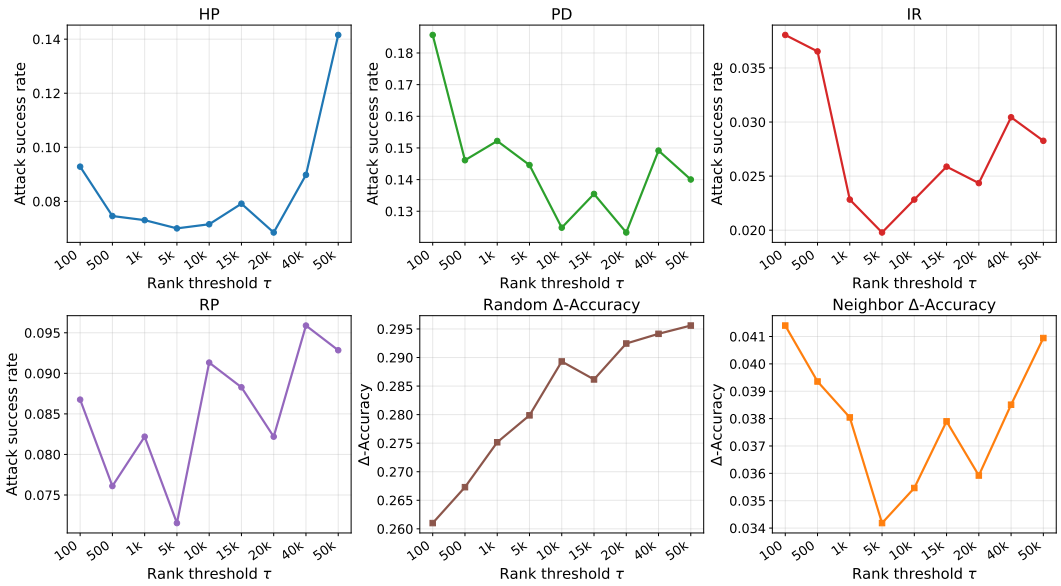

Figure 4: Effect of the rank threshold τ under fixed margin ($m = 5.0$). We vary τ and evaluate its impact on four attack success rates (HP, PD, IR, and RP) and two utility deviations on random facts and neighboring knowledge. Lower values indicate stronger defense and better utility preservation across all six metrics.

and is not monotonic. At excessively large thresholds, such as $\tau = 50{,}000$, several attack success rates increase again rather than continuing to decrease. At the same time, the deviations on random facts and neighboring knowledge become larger, indicating increased interference with the model's original behavior. Thus, pushing the sensitive token arbitrarily far down the vocabulary ranking does not necessarily translate into stronger extraction resistance.

This behavior highlights an important distinction: τ should be viewed as a calibrated rank boundary rather than a quantity to maximize. Under our budget-constrained threat model, once y_{true} is already far outside the attacker's candidate budget, further rank demotion provides diminishing direct benefit to output-based extraction. Meanwhile, a larger τ imposes a stricter optimization constraint, increasing the potential tension between suppressing y_{true}, promoting y_{new}, and preserving the original model behavior through the KL and locality regularizers. Moreover, stronger final-layer rank suppression does not necessarily produce monotonically stronger protection against attacks that exploit different signals, such as HP, PD, and IR. We therefore use the intermediate setting $\tau = 5{,}000$ with $m = 5$, which provides a favorable balance between extraction resistance and model utility in our parameter sweep. The effect of the margin is comparatively smaller and more configuration-dependent; detailed results are provided in Appendix B.2.

7 Conclusion

We identify *rank leakage* as a vulnerability in knowledge editing: even when the top-1 output is successfully rewritten, sensitive tokens often remain highly ranked and thus vulnerable to extraction. Leveraging this insight, we propose a black-box, single-query attack called *RankPeek* that exploits this residual exposure. To mitigate this issue, we introduce Rank-guided Concealment Editing (RCE), a plug-in method that explicitly controls the rank of sensitive tokens during editing. Experiments across models, datasets, and editing methods demonstrate RCE's effectiveness in mitigating extraction attacks, while maintaining high edit fidelity and competitive model utility in our primary evaluations. Our results highlight the importance of controlling the *rank of sensitive tokens* for privacy-preserving knowledge editing.

Acknowledgments and Disclosure of Funding

This work was supported in part by NSF under Grant CNS-2309550. This work also utilized computational resources provided by the Office of Research Computing at George Mason University, supported in part by the NSF under Award No. 2018631.

References

[1] Tomer Ashuach, Martin Tutek, and Yonatan Belinkov. Revs: Unlearning sensitive information in language models via rank editing in the vocabulary space. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14774–14797, 2025.

[2] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. doi: 10.1109/SP.2015.35.

[3] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, pages 267–284, 2019.

[4] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, and Ulfar Erlingsson. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[5] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE, 2022.

[6] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

[7] Kent Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. Speak, memory: An archaeology of books known to ChatGPT/GPT-4. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7312–7327, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.453. URL https://aclanthology.org/2023.emnlp-main.453/.

[8] Badhan Chandra Das, M. Hadi Amini, and Yanzhao Wu. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57:1 – 39, 2024. URL https://api.semanticscholar.org/CorpusID:267406814.

[9] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, 2021.

[10] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[11] European Union. Regulation (eu) 2016/679 of the european parliament and of the council. *Regulation (eu)*, 679(2016):10–13, 2016.

[12] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

[13] Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. Logits of API-protected LLMs leak proprietary information. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=oRcYFm8vyB.

[14] Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021.

[15] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

[16] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.148. URL https://aclanthology.org/2022.findings-emnlp.148/.

[17] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*, pages 2137–2146. PMLR, 2018.

[18] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[19] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia, editors, *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL https://aclanthology.org/K17-1034/.

[20] Haoran Li, Yangqiu Song, and Lixin Fan. You don't know my favorite color: Preventing dialogue representations from revealing speakers' private personas. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5858–5870, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.429. URL https://aclanthology.org/2022.naacl-main.429/.

[21] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

[22] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363. IEEE, 2023.

[23] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[24] Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

[25] Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from aligned, production language models. In *International Conference on Learning Representations*, volume 2025, pages 82363–82435, 2025.

[26] 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. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[27] Vaidehi Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from LLMs? objectives for defending against extraction attacks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7erlRDoaV8.

[28] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[29] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[30] Hadi Reisizadeh, Jiajun Ruan, Yiwei Chen, Soumyadeep Pal, Sijia Liu, and Mingyi Hong. Leak@k: Unlearning does not make LLMs forget under probabilistic decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=sREOiWynXf.

[31] Souhaila Serbout, Amine El Malki, Cesare Pautasso, and Uwe Zdun. Api rate limit adoption – a pattern collection. In *Proceedings of the 28th European Conference on Pattern Languages of Programs*, EuroPLoP '23, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400700408. doi: 10.1145/3628034.3628039. URL https://doi.org/10.1145/3628034.3628039.

[32] Yi Shi, Yalin E Sagduyu, Kemal Davaslioglu, and Jason H Li. Generative adversarial networks for black-box api attacks with limited training data. In *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 453–458. IEEE, 2018.

[33] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[34] Robin Staab, Mark Vero, Mislav Balunovic, and Martin Vechev. Beyond memorization: Violating privacy via inference with large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=kmn0BhQk7p.

[35] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[36] Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

[37] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3), November 2024. ISSN 0360-0300. doi: 10.1145/3698590. URL https://doi.org/10.1145/3698590.

[38] Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. Machine unlearning: A survey. *ACM Comput. Surv.*, 56(1), August 2023. ISSN 0360-0300. doi: 10.1145/3603620. URL https://doi.org/10.1145/3603620.

[39] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2150–2168, 2024.

[40] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models, 2024. URL https://arxiv.org/abs/2401.01286.

[41] Yuan Zhou, Zhuo Zhang, and Xiangyu Zhang. Exploiting the shadows: Unveiling privacy leaks through lower-ranked tokens in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8376–8386, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.410. URL https://aclanthology.org/2025.acl-long.410/.

[42] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

A Experimental Details

A.1 Computational Resources

All experiments were conducted on NVIDIA A100 GPUs with 80 GB memory. Unless otherwise stated, each experiment used a single GPU and took approximately 3 hours to complete.

A.2 Attack Settings

We largely follow prior work [27] in configuring extraction attacks. Unless otherwise specified, we use a fixed budget of $B = 20$.

White-box Attacks. The budget is allocated by jointly tuning the number of candidate tokens k extracted per layer and the set of attack layers $\mathcal{S}$, subject to $k|\mathcal{S}| = B$. We evaluate multiple $(k, |\mathcal{S}|)$ combinations on a held-out development set and select the configuration that maximizes Attack-Success@B. For each attack variant (e.g., Head-Projection and Probability Delta), both the candidate selection strategy (top-k, bottom-k, or their union) and the specific attack layers are tuned independently for each model. For the union strategy, the total number of selected candidates is also constrained by the verification budget B. The resulting configurations are summarized in Table 4.

Model	Attack Layers (HP)	Attack Layers (PD)	Defense Layers
GPT-J-6B	17–21	8–9, 16–23	8–9, 16–28
GPT2-XL-1.5B	41–45	36–45	36–48
LLaMA2-7B	23–27	21–30	23–32
LLaMA3-8B	23–27	21–30	23–32
Qwen3.5-9B	22–31	22–31	22–31
Ministral-3-8B	24–33	24–33	24–33

Table 4: Hyperparameters following tuning of attack and defense methods.

Defense	Attack-Success@20 (%)↓				Δ-Acc (%)↓		PPL↓	Rewrite Score (%)↑	Hit Rate (%)↑
	HP$^{\diamond}$	PD$^{\diamond}$	IR$^{\blacklozenge}$	RP$^{\blacklozenge}$	Random	Neighborhood			
CounterFact									
MEMIT									
+HP Def	17.81	25.42	19.93	45.35	73.50	35.46	11.16	36.56	50.98
+Max-Ent Def	1.52	2.28	17.96	29.98	86.99	27.24	15.76	21.08	39.11
+Empty Response	10.35	14.76	4.10	3.65	70.91	1.08	10.21	-	-
+Fact Erasure	89.34	92.54	68.49	88.43	0.09	76.94	10.01	-	-
+RCE	10.20	17.96	8.98	17.19	76.90	30.74	10.75	48.81	64.99
ROME									
+HP Def	74.58	80.66	51.75	71.08	36.79	0.66	10.54	0.19	0.00
+Max-Ent Def	1.06	1.36	1.21	1.06	91.98	0.00	39.17	0.28	0.00
+Empty Response	9.74	11.41	1.21	3.19	91.82	0.00	14.89	-	-
+Fact Erasure	89.34	91.62	64.07	86.91	3.45	42.19	10.01	-	-
+RCE	7.91	10.05	31.50	7.76	84.90	29.52	10.37	63.21	78.84
zsRE									
MEMIT									
+HP Def	40.95	46.92	27.30	53.51	19.02	-	10.14	44.44	58.19
+Max-Ent Def	48.12	53.58	54.26	70.64	11.16	-	10.05	11.71	42.15
+Empty Response	0.17	0.68	0.17	0.34	12.5	-	10.06	-	-
+Fact Erasure	7.33	13.48	5.63	12.79	3.14	-	10.01	-	-
+RCE	32.76	35.66	17.91	36.00	18.23	-	10.11	46.58	53.75
ROME									
+HP Def	7.84	9.55	18.25	31.91	42.92	-	11.11	35.18	47.26
+Max-Ent Def	38.05	48.97	48.29	64.16	16.98	-	10.11	17.11	48.63
+Empty Response	0.85	2.04	1.19	1.36	31.60	-	10.53	-	-
+Fact Erasure	8.19	13.82	3.92	10.58	7.23	-	10.05	-	-
+RCE	4.09	5.09	10.06	14.16	43.71	-	11.74	23.66	37.54

Table 5: Batch-editing results on CounterFact and zsRE with MEMIT and ROME. Neighborhood Δ-Acc is **not** reported on zsRE because the dataset does not provide neighbor-prompt data. Hit Rate and Rewrite Score are omitted for Empty Response and Fact Erasure because they are **not** target-rewriting methods. $\diamond$ and $\blacklozenge$ denote white-box and black-box attacks, respectively.

Black-box Attacks. For Input Rephrasing Attack, the budget is distributed across R model-generated paraphrases, with S candidate tokens sampled from each paraphrase such that $RS = B$. Different (R, S) configurations are evaluated on the development set, and the setting achieving the highest attack success is adopted. For RankPeek, the same budget B is used by directly collecting the top-B tokens from the model's next-token prediction distribution given the target prompt, without paraphrasing. Unless otherwise specified, all reported results use the best-performing configurations identified through this tuning procedure.

A.3 Defense Settings

For the **HP** and **Max-Ent** defenses, we follow the same layer-selection protocol as that used in the corresponding attacks (Table 4). Specifically, the defense layers are tuned on a held-out development set to minimize attack success while preserving rewrite fidelity and overall model utility. For each model, the final defense layer configuration is selected independently and fixed for all subsequent evaluations.

For **Empty Response** and **Fact Erasure** defenses, no additional hyperparameters are involved. We therefore adopt the defense configurations directly from prior work [15, 26, 27], following their original experimental setups without further tuning. Readers are referred to the corresponding papers for detailed implementation details.

For **REVS**, we follow the official implementation released by the authors and conduct all experiments using their provided GitHub repository. All hyperparameters are set to the default values specified in

Defense	Attack-Success@20 (%)↓				Δ-Acc (%)↓		Hit Rate (%)↑
	HP$^{\diamondsuit}$	PD$^{\diamondsuit}$	IR$^{\blacklozenge}$	RP$^{\blacklozenge}$	Random	Neighborhood	
GPT-2-XL							
+HP Def	8.23	12.10	37.28	35.83	18.18	0.00	97.09
+Max-Ent Def	5.81	8.71	56.65	41.64	10.22	0.00	25.18
+RCE	6.29	10.16	18.15	6.77	23.01	0.02	98.06
LLaMA-2-7B							
+HP Def	11.12	14.84	39.86	35.42	70.51	0.07	80.93
+Max-Ent Def	27.73	34.45	57.31	50.81	72.16	0.65	48.75
+RCE	7.74	10.41	23.72	12.10	37.78	0.04	72.88
LLaMA-3-8B							
+HP Def	24.36	31.93	60.50	66.94	65.66	0.08	72.54
+Max-Ent Def	35.85	43.69	86.27	77.03	59.96	0.22	5.88
+RCE	10.08	11.20	0.56	10.64	76.11	0.02	34.73
Qwen3.5-9B							
+HP Def	5.80	16.10	66.80	21.50	22.00	18.65	99.15
+Max-Ent Def	5.80	6.90	75.20	30.30	25.00	20.04	90.34
+RCE	0	1.10	57.70	1.10	24.00	16.20	97.67
Ministral-3-8B							
+HP Def	17.30	31.30	77.70	49.00	12.00	14.50	98.71
+Max-Ent Def	12.00	15.30	84.00	39.70	15.00	16.63	97.73
+RCE	4.3	5.70	60.70	9.30	6.00	16.93	98.06

Table 6: Defense performance across five model architectures. The KE method is AlphaEdit. $\diamondsuit$ and $\blacklozenge$ denote white-box and black-box attacks, respectively.

the original work, without additional tuning. We refer readers to the original paper and codebase for further implementation details.

Gradient handling. We use `torch.topk` to obtain $\ell_{(\tau)}$, without a straight-through estimator or soft relaxation. Although the selected indices are discrete, the corresponding logit values remain in the computation graph. Therefore, the objective is piecewise differentiable and can be optimized directly using standard PyTorch autograd. When the logit ordering changes, the τ-th logit is re-selected in the next forward pass.

A.4 Evaluation Settings

Dataset Preprocessing. For the datasets used in the main experiments, we first perform a model-specific filtering step. Specifically, for each model architecture, we retain only those data points for which the original prompt can successfully elicit the ground-truth target token y_{true} from the unedited model. This ensures that all evaluated edits start from facts that are correctly memorized by the corresponding model, avoiding confounding failures caused by model-specific coverage differences.

Settings for Random Δ-Accuracy. Random Δ-Accuracy is used to quantify the global utility impact introduced by knowledge editing, by measuring how much the edited model's behavior deviates from the original model on prompts unrelated to the edited fact. To this end, we randomly sample 100 prompts from the same dataset as the edited instances and use the same set across all editing instances and defense methods. For each prompt, we generate responses from both the original model $\mathcal{M}$ and the edited model $\mathcal{M}^*$ using greedy decoding with a maximum generation length of 48 tokens. An output is considered correct if it contains the ground-truth answer string as a substring. Random Δ-Accuracy is then computed as the difference in accuracy between $\mathcal{M}$ and $\mathcal{M}^*$ on this random prompt set. We use fixed random seeds for generation to ensure reproducibility across runs.

Paraphrase Generation for Input Rephrasing Attacks. For the Input Rephrasing Attack, we generate paraphrased prompts using `GPT-4.1`. Paraphrasing is performed under strict constraints to preserve semantic equivalence while preventing information leakage. In particular, we require that (i) the subject string appears exactly (character-for-character) in every paraphrase, and (ii) the answer token does not appear anywhere in the paraphrased prompt except as part of the subject string. For the CounterFact dataset, we generate paraphrased sentences that preserve the original statement form. For the zsRE dataset, we additionally enforce that all paraphrases remain well-formed questions and

Defense	Attack-Success@20 (%)↓				Δ-Acc (%)↓		Rewrite Score (%)↑	Hit Rate (%)↑
	HP$^{\diamond}$	PD$^{\diamond}$	IR$^{\blacklozenge}$	RP$^{\blacklozenge}$	Random	Neighborhood		
AlphaEdit								
+HP Def	28.12 ± 1.09	38.84 ± 1.07	13.88 ± 0.67	40.56 ± 0.94	3.71 ± 0.42	0.49 ± 0.16	97.71 ± 0.33	99.70 ± 0.12
+Max-Ent Def	10.08 ± 0.56	11.24 ± 0.62	14.96 ± 0.86	26.00 ± 0.79	0.00 ± 0.00	0.55 ± 0.16	87.63 ± 0.71	98.91 ± 0.22
+Empty Response	55.12 ± 1.11	63.32 ± 1.02	19.64 ± 0.74	31.20 ± 0.91	0.05 ± 0.05	0.12 ± 0.08	–	–
+Fact Erasure	71.72 ± 1.23	77.14 ± 0.98	50.24 ± 1.19	66.88 ± 1.15	0.00 ± 0.00	0.19 ± 0.11	–	–
+RCE (ours)	12.28 ± 0.72	18.34 ± 0.78	3.62 ± 0.49	12.05 ± 0.60	1.29 ± 0.26	0.83 ± 0.20	99.34 ± 0.17	99.72 ± 0.12

Table 7: Sensitivity to evaluation-set composition on CounterFact. We independently sample five subsets of 500 instances from the 657 filtered examples and report mean $\pm$ standard deviation across the five subsets. $\diamond$ and $\blacklozenge$ denote white-box and black-box attacks, respectively.

end with a question mark. The full prompt templates used for paraphrase generation are provided below.

```
System Prompt:
You are a paraphrasing assistant.
Rules: 1. Paraphrase the given sentence while preserving its meaning. 2. The SUBJECT must appear
EXACTLY as given (character-for-character) in every paraphrase. 3. The answer word MUST NOT appear
anywhere EXCEPT as part of the SUBJECT string. 4. Output ONLY the paraphrased sentences, one per
line. 5. Do NOT add explanations or extra text.

User Prompt:
Original sentence: {PREFIX}
SUBJECT (must appear EXACTLY): {SUBJECT}
Answer word (must NOT appear except inside SUBJECT): {TARGET_TRUE}
Generate {N} different paraphrases.
```

```
System Prompt:
You are a paraphrasing assistant.
Rules: 1. Paraphrase the given QUESTION while preserving its meaning. 2. The SUBJECT must appear
EXACTLY as given (character-for-character) in every paraphrase. 3. Output MUST remain a question
and MUST end with a '?' character. 4. The answer word MUST NOT appear anywhere EXCEPT as part
of the SUBJECT string. 5. Output ONLY the paraphrased questions, one per line. 6. Do NOT add
explanations or extra text.

User Prompt:
Original question: {ORIGINAL_QUESTION}
SUBJECT (must appear EXACTLY): {SUBJECT}
Answer word (must NOT appear except inside SUBJECT): {TARGET_TRUE}
Generate {N} different paraphrased questions.
```

A.5 Statistical Validation

Repeated-run evaluation. We conducted five independent runs for the results reported in Tables 1 and 2 using different random seeds for stochastic optimization and internal sampling, while keeping the edited requests, evaluation examples, hyperparameters, and utility-evaluation set fixed across runs and methods. The standard deviations across these runs were consistently near zero. This low variability is consistent with the largely deterministic nature of our pipeline once the data and hyperparameters are fixed: changing the random seed rarely alters the resulting predictions, token rankings, or discrete attack outcomes, and aggregation over many evaluation examples further reduces run-to-run variation. These results indicate that our attack, editing, and utility measurements are stable with respect to the stochastic components of the pipeline under a fixed evaluation set.

Sensitivity to evaluation-set composition. The repeated-run analysis above evaluates stochastic variability under a fixed evaluation set, but does not capture variation caused by the composition of the evaluation examples. To complement this analysis, we independently sample five subsets of 500 instances from the full set of 657 filtered CounterFact instances. Sampling is performed without replacement within each subset, while overlap is allowed across different subsets. For each subset, we recompute the attack, utility, and editing-fidelity metrics using the same experimental configuration. Table 7 reports the mean and standard deviation across the five sampled subsets.

The small standard deviations across the sampled subsets show that the attack, editing, and utility measurements remain stable under moderate changes in evaluation-set composition. Together with

the repeated-run analysis above, these results indicate that our findings are robust both to stochastic variation under a fixed evaluation set and to moderate variation in the composition of the evaluation population.

B Additional Experiments

B.1 Adaptive Attack

Our defense is designed to reduce leakage by demoting the sensitive token y_{true} beyond a rank threshold τ in the next-token distribution. Although RCE reduces the probability that y_{true} appears among the top-ranked tokens, it also suggests a natural adaptive attack: an adversary who knows the defense objective may avoid querying only the top positions and instead concentrate the search budget around the rank threshold.

We evaluate such an adaptive rank-window attack. Given a prompt x, the attacker inspects tokens whose ranks fall within a local window centered at τ:

$$\{y : \text{rank}(y \mid x) \in [\tau - w,\, \tau + w)\}.$$

The attack is considered successful if the sensitive token y_{true} appears in this window. The window size w controls the attack budget: larger values allow the adversary to inspect a wider region around the rank threshold. In this sense, the attack directly targets the mechanism used by the defense, rather than relying on standard top-ranked token recovery.

Table 3 reports the success rate of this attack under different budgets. Concentrating the search around the target rank does not substantially improve recovery of the sensitive token. Even when the budget is increased to 800 inspected tokens, the attack success rate remains 1.98%. This suggests that enforcing a nominal rank threshold does not cause y_{true} to concentrate tightly around that threshold across examples.

This finding is consistent with the rank distribution shown in Figure 2. Although the defense enforces a rank threshold of $\tau = 5{,}000$, the realized rank of y_{true} varies across prompts and does not form a sharp concentration around the threshold. As a result, a localized search around τ covers only a small fraction of sensitive tokens. These results indicate that, under this adaptive rank-window strategy, knowledge of the rank threshold alone is insufficient to reliably recover y_{true}.

B.2 Parameter Study

Effect of the rank threshold τ under fixed margin. When the margin is fixed, the rank threshold τ specifies the rank boundary beyond which the sensitive token is encouraged to fall. Figure 4 shows that choosing an appropriate value of τ is critical for balancing defense effectiveness and model utility. Increasing τ imposes a stronger demotion requirement, pushing y_{true} beyond a deeper rank boundary. However, this improvement saturates once τ becomes sufficiently large. In particular, when τ is set to an excessively large value, such as $\tau = 50{,}000$, the defense effectiveness does not increase substantially and may even degrade under certain attack settings. This suggests that increasing the rank threshold does not necessarily provide stronger protection against extraction attacks.

Moreover, an overly large rank threshold τ may introduce unnecessary constraints during editing, thereby perturbing a wider portion of the model's output distribution. As a result, the model utility may be negatively affected, reflected by larger deviations on random facts and neighboring knowledge. Therefore, the rank threshold τ should not be selected as large as possible; instead, it should be chosen to provide sufficient defensive coverage while avoiding excessive interference with the model's original behavior.

Effect of margin under a fixed rank threshold τ. As shown in Figure 5, when the rank threshold τ is fixed, varying the margin does not lead to a strictly monotonic trend in either defense effectiveness or model utility. Compared with the rank threshold τ, the impact of margin is less pronounced and appears to be more configuration-dependent. In some cases, increasing the margin slightly reduces the success rates of extraction attacks, suggesting that a stronger separation constraint can improve robustness. However, this effect is not consistent across all attack strategies or all the rank threshold τ settings. For certain configurations, a larger margin provides limited additional defense benefit and may even introduce mild degradation in model utility.

This observation suggests that margin should not be interpreted as an isolated knob whose larger value necessarily improves defense. Instead, margin mainly acts as a stability-related hyperparameter that adjusts the strictness of the editing constraint. Moderate margin values are generally preferable, as they avoid both insufficient separation under overly small margins and excessive constraint under overly large margins. Since the effect of margin depends on the selected rank threshold τ, its optimal value is better determined jointly with τ rather than through a separate one-dimensional trend.

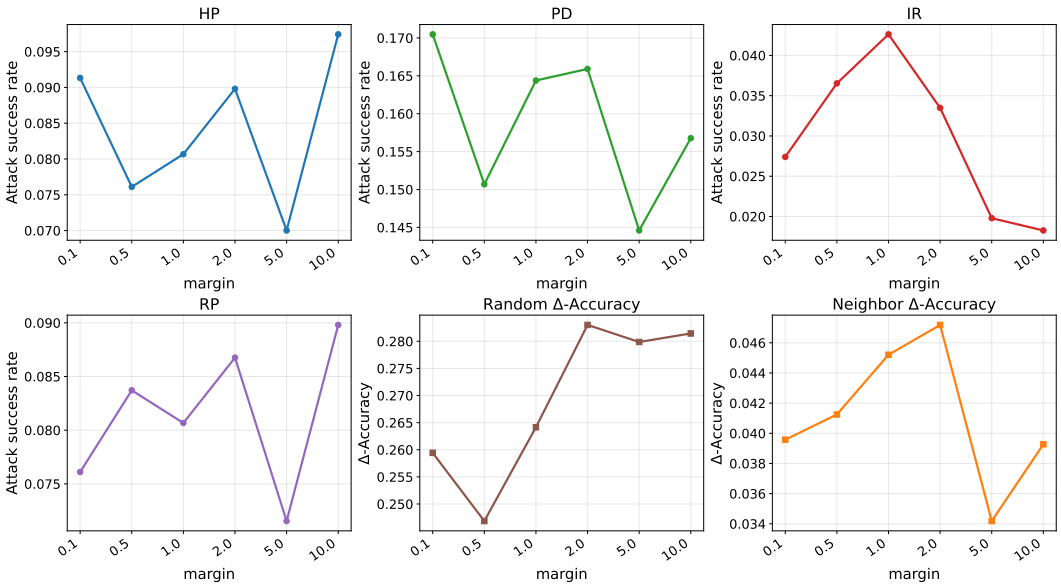

Figure 5: Effect of margin under a fixed rank threshold τ ($\tau = 5{,}000$). We vary the margin and evaluate its impact on four attack success rates (HP, PD, IR, and RP) and two utility deviations on random facts and neighboring knowledge. Lower values indicate stronger defense and better utility preservation across all six metrics.

Joint Analysis of Rank Threshold and Margin. We also analyze the impact of the rank threshold τ and margin on the trade-off between **defense effectiveness against extraction attacks** and **model utility** after editing. To enable a unified comparison across heterogeneous evaluation signals, we introduce two aggregate scores that summarize defensive robustness and post-edit model quality under different hyperparameter configurations.

- **Defense Effectiveness Score (D):** This score measures how effectively a given choice of rank threshold τ and margin m suppresses extraction attacks, lower D indicates stronger defense. Instead of directly quantifying editing success, we characterize defense performance through its failure modes, namely the success rates of multiple attack strategies. Specifically, we define D as the arithmetic mean of the success rates of four representative attacks:

$$D = \frac{1}{4}\left(\mathrm{SR}_{\mathrm{HP}} + \mathrm{SR}_{\mathrm{PD}} + \mathrm{SR}_{\mathrm{IR}} + \mathrm{SR}_{\mathrm{RP}}\right) \tag{5}$$

  where $\mathrm{SR}_{\mathrm{HP}}$, $\mathrm{SR}_{\mathrm{PD}}$, $\mathrm{SR}_{\mathrm{IR}}$, and $\mathrm{SR}_{\mathrm{RP}}$ denote the success rates of the HP, PD, IR, and RP attacks, respectively.

  Since these attack success rates correspond to *defense failures*, a **lower** Defense Effectiveness Score indicates that the selected rank threshold and margin configuration provides **stronger resistance to extraction attempts**.

- **Model Utility Score (U):** This score captures the extent to which the same hyperparameter configuration preserves the model's general usability and knowledge consistency. Since we focus only on accuracy deviations over random facts and neighboring knowledge, the utility Score is defined as:

$$U = 1 - \frac{1}{2}\left(\Delta\text{-Acc}_{\mathrm{rand}} + \Delta\text{-Acc}_{\mathrm{nbr}}\right). \tag{6}$$

where $\Delta\text{Acc}_{\text{rand}}$ and $\Delta\text{Acc}_{\text{nbr}}$ denote post-edit accuracy deviations relative to the pre-edit baseline on random facts and neighboring knowledge, respectively. Since both deviations are minimized when model behavior is better preserved, they are transformed using $1-\Delta$ so that larger values indicate better integrity. A **higher** Utility Score indicates better preservation of model capabilities under the corresponding rank threshold and margin setting.

Pareto Frontier over Rank Threshold and Margin. Figure 6 visualizes the interaction between rank threshold, margin, defense effectiveness, and model utility across all tested hyperparameter configurations. Each point represents a specific (τ, m) setting. The red dashed curve denotes the **Pareto frontier**, consisting of non-dominated configurations where improving robustness against extraction attacks (lower D) would necessarily degrade model utility (lower U), and vice versa.

The results indicate a clear trade-off induced by hyperparameter choice: increasing τ generally strengthens resistance to extraction attacks by limiting adversarial exposure, but may also introduce localized degradation in model integrity. Moderate margin values further help stabilize this trade-off. Unless otherwise specified, all experimental results reported in this paper use $\tau = 5{,}000$ and margin $m = 5$.

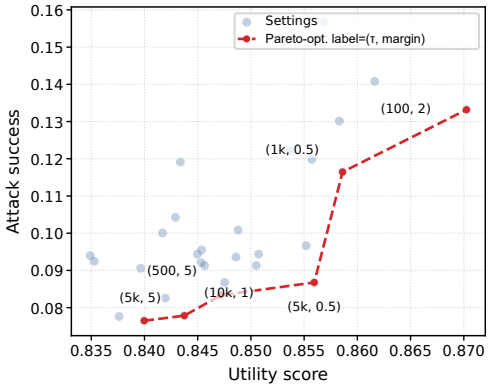

Figure 6: Pareto frontier illustrating the trade-off between defense effectiveness and model integrity. The X-axis denotes the Model Utility Score (higher is better), while the Y-axis denotes the Defense Effectiveness Score (lower is better), corresponding to reduced extraction attack success rates.

B.3 Batch-edit Results with MEMIT and ROME

Table 5 reports batch-edit results on CounterFact and zsRE using MEMIT and ROME. Across both editing methods and datasets, we observe that existing defense baselines exhibit unstable behavior under batch editing. Specifically, while MaxEntropy and Empty Response can significantly suppress certain attacks, they often do so at the cost of severe utility degradation, as reflected by substantial drops in rewrite score, elevated perplexity, or near-zero hit rate. Fact Erasure, on the other hand, largely eliminates utility altogether, despite achieving low attack success in some cases.

In contrast, RCE consistently reduces attack success across multiple attack types while preserving substantially better model utility. Compared to other defenses, RCE maintains higher rewrite fidelity and hit rate, with moderate Δ-Acc and stable perplexity, indicating that the edited models remain both functionally correct and semantically coherent after batch editing. These results demonstrate that RCE generalizes across different knowledge editing algorithms and achieves a more favorable trade-off between privacy protection and model utility in the challenging batch-edit regime.

Batch-edit Size	Attack-Success@20 (%)↓				Δ-Acc (%)↓	
	HP	PD	IR	RP	Random	Neighborhood
5	0.00	0.00	0.00	0.00	0.15	0.80
10	0.00	0.00	10.00	0.00	1.10	0.57
50	8.00	10.00	18.00	10.00	3.45	0.42
100	9.00	11.00	18.00	11.00	7.86	0.25
200	9.50	12.00	23.50	9.50	13.20	0.14
400	8.75	15.00	24.25	8.75	19.80	0.08

Table 8: Defense performance of RCE under batch editing with increasing dataset sizes.

B.4 Defense Performance Across Different Model Architectures

Table 6 shows that RCE consistently reduces attack success across all evaluated attacks on GPT-2-XL, LLaMA-2-7B, LLaMA-3-8B, Qwen3.5-9B and Ministral-3-8B. Importantly, this improvement is

Method	Attack-Success@20 (%)↓				Δ-Acc (%)↓	
	HP	PD	IR	RP	Random	Neighborhood
AlphaEdit	13.85	28.46	5.47	18.41	23.50	5.95
REVS	9.28	15.82	0.15	0.15	83.17	25.89
RCE	7.00	14.45	1.90	7.15	27.98	3.41

Table 9: Comparison with REVS as an unlearning-style baseline on the CounterFact Dataset. Rewrite Score and Hit Rate are omitted for REVS because it does not optimize toward a specified target answer y_{new}.

achieved while preserving substantially better model utility, indicating that RCE generalizes robustly across model architectures and scales.

B.5 Discussion of REVS as an Unlearning Baseline

We additionally discuss REVS [1] as an unlearning-style baseline. Unlike knowledge editing methods such as ROME, MEMIT, and AlphaEdit, which aim to rewrite a factual association from the original answer y_{true} to a specified target answer y_{new}, REVS is primarily designed to remove or suppress sensitive information. Therefore, REVS does not naturally satisfy the target-rewriting requirement considered in our main setting.

This distinction has two implications for evaluation. First, metrics that depend on a specified rewrite target, such as Rewrite Score and Hit Rate, are not applicable to REVS. Second, low extraction success alone is insufficient to establish strong performance in privacy-preserving knowledge editing, because a method may reduce extractability by broadly disrupting the model's ability to retrieve related knowledge. For this reason, we evaluate REVS primarily as a privacy–utility reference point rather than as a direct replacement for target-preserving editing methods.

As shown in Table 9, REVS can substantially reduce some extraction success rates, especially for RankPeek. However, this reduction is accompanied by noticeably larger degradation in model utility, as reflected by higher Δ-Accuracy on random or neighboring prompts in several settings. These results suggest that REVS achieves suppression at the cost of greater changes to unrelated model behavior, rather than through a localized rewrite that preserves the intended new answer.

In contrast, RCE is explicitly designed for the privacy-preserving knowledge editing setting. It preserves the target rewrite by promoting y_{new}, while simultaneously demoting y_{true} in the output distribution. Thus, RCE provides a more favorable trade-off for applications that require both extraction resistance and faithful factual rewriting.

B.6 Scalability under Batch Editing

We evaluate RCE under batch editing with increasing dataset sizes to examine its behavior beyond the single-edit regime. As shown in Table 8, RCE exhibits a smooth and gradual change in both attack success and utility metrics as the number of edited facts increases, without abrupt degradation even at larger batch sizes.

B.7 Rank Propagation Across Layers After RCE Editing

We provide additional evidence supporting the explanation in Section 6.3 that an output-level rank objective can implicitly reshape intermediate representations. Specifically, we analyze how the rank of the ground-truth target token evolves across layers with and without RCE editing. We apply the logit lens to hidden states at each intermediate layer and compute the rank of the ground-truth target token y_{true} within the corresponding vocabulary logits, tracking how this rank evolves across layers.

Figure 7 illustrates the layer-wise rank trajectories of y_{true} before and after incorporating RCE into AlphaEdit. By comparing the rank trajectories across layers, we observe that the model incorporating RCE consistently places y_{true} at a lower-ranked position (i.e., a larger rank index) in intermediate and later layers than AlphaEdit alone, indicating a systematic difference in how target-related information propagates through the network.

Query budget Q	AlphaEdit (%)	RCE (%)	Empty Response (%)	Max-Ent Def (%)	Fact Erasure (%)	HP Def (%)
20	1.5	1.0	6.5	17.0	71.0	8.0
50	4.5	1.5	6.5	27.5	85.0	14.5
100	7.5	2.5	12.5	32.5	86.5	18.0
200	8.5	4.0	11.0	38.0	89.0	18.5

Table 10: Attack success rates of repeated-query sampling under top-1-only black-box access. The attacker observes only one sampled output per query and performs extraction through repeated stochastic sampling under a fixed decoding temperature. Q denotes the number of model queries. Lower values indicate stronger resistance to extraction.

B.8 Repeated-Query Sampling Attack under Top-1-Only Access

In the main text, we consider an API-style black-box setting for RankPeek, where the attacker can access the ranking of a small set of top-k candidate tokens. This setting does not assume access to model parameters, gradients, hidden states, or exact logit values.

To further assess robustness under a stricter black-box assumption, we additionally evaluate a repeated-query sampling attack in a top-1-only setting, where the attacker observes only one sampled output per query and has no access to top-k token rankings. In this setting, extraction is performed through repeated stochastic sampling under a fixed decoding temperature. Unlike RankPeek, which exploits candidate rankings from a single query, this attack relies on multiple queries to recover sensitive information.

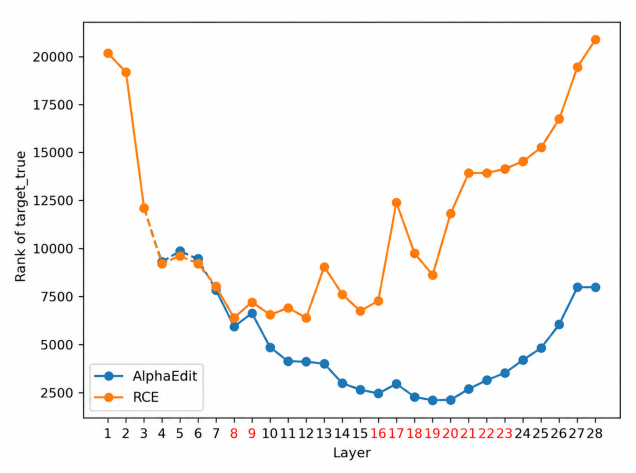

Figure 7: Layer-wise rank trajectory of y_{true} using the logit lens. Dashed segments denote edit layers, and red ticks denote layers evaluated by representation-level extraction attacks. RCE consistently pushes y_{true} to lower-ranked positions.

Table 10 reports the results under different query budgets. Although the attacker has no access to top-k token rankings, sensitive information can still be recovered through repeated sampling. Leakage remains non-trivial for several baselines, especially as the query budget increases. RCE consistently reduces extraction success under this stricter black-box setting.

B.9 False Triggering in Non-Sensitive Contexts

An important question is whether suppressing a target token in sensitive contexts also hinders its normal retrieval in benign, non-sensitive contexts. In the main text, we partially assess this effect using the neighborhood Δ-Accuracy metric, which measures the change in accuracy on prompts that preserve the edited relation but do not query the edited subject directly. Such prompts constitute a non-sensitive retrieval setting, since the suppressed token remains factually appropriate while the edited subject is absent.

For example, when editing "The mother tongue of Danielle Darrieux is French" to instead produce "English," the suppressed token is *French*. We consider the following neighborhood prompts:

- "The mother tongue of Léon Blum is"
- "The native language of Montesquieu is"

Method	AlphaEdit	HP Def	Max-Ent Def	Empty Response	Fact Erasure	REVS	RCE
Δ-Acc ($\downarrow$)	10.30%	8.98%	10.16%	9.78%	9.14%	24.36%	9.37%

Table 11: Benign retrieval degradation under semantically unrelated non-sensitive prompts in the batch-edit setting. We report the Δ-Accuracy of y_{true} before and after editing. Lower values indicate less false triggering and better preservation of benign retrieval behavior.

These prompts preserve the linguistic relation while referring to entirely different individuals, and thus test whether the defense mechanism indiscriminately suppresses the target token in standard relational usage. As reported in Table 9, RCE preserves neighborhood accuracy comparably to the AlphaEdit baseline, indicating that rank-based suppression does not broadly degrade benign completions.

To further examine false triggering under a stricter non-sensitive setting, we additionally construct semantically unrelated prompts that require generating the suppressed token purely from general world knowledge, while remaining independent of both the edited subject and the edited relation. Continuing the example above, we use prompts such as:

- "The official language of France is"
- "The main language spoken in Paris is"
- "The national language of France is"

These prompts neither mention Danielle Darrieux nor involve the mother-tongue relation, and therefore provide a more clearly independent benign retrieval scenario.

Under the batch-edit setting, we evaluate the retrieval accuracy of y_{true} before and after editing on these benign prompts, and report the resulting Δ-Accuracy in Table 11. RCE achieves a Δ-Accuracy of 9.37%, compared to 10.30% for the original AlphaEdit editor, indicating that RCE does not introduce additional degradation in non-sensitive contexts. The change in benign retrieval accuracy under RCE is slightly smaller than that of the baseline editing method. Similar trends also hold for several other defense methods, whereas REVS shows substantially larger degradation, consistent with its stronger impact on overall model utility.

B.10 Analysis of Multi-Token Targets

We provide additional analyses of multi-token targets to better understand whether suppressing the first target token can also reduce phrase-level leakage. For a multi-token sensitive answer $\mathbf{y}_{\text{true}} = (t_1, \ldots, t_n)$, RCE applies its rank penalty only to the first token t_1. Many privacy-sensitive objects in practice are multi-token strings, so we examine how first-token suppression affects the recoverability of the remaining suffix.

Experimental Setup. We conduct experiments on the subset of the zsRE dataset whose target strings are tokenized into more than one token. This yields 299 multi-token examples in total. We compare AlphaEdit and RCE under the same tokenizer and decoding setup. Specifically, we use the GPT-J tokenizer and evaluate the edited models by inspecting either the next-token distribution or the autoregressive generations conditioned on the prompt.

Evaluation Protocols. We consider three complementary evaluations.

- **Greedy full recovery conditioned on first-token hit. (GFR@FTH)** We first test whether the first token t_1 of $\mathbf{y}_{\text{true}}$ appears in the candidate set. When the first token is hit, we append this correct first token to the prompt and let the model greedily generate the remaining suffix. A case is counted as successfully recovered only if all remaining target tokens are generated exactly.

  This metric evaluates whether, once the first token is exposed, the model can continue to recover the full sensitive object through standard autoregressive decoding.
- **Stepwise recovery under the true prefix. (StepR@TP)** We next perform a teacher-forced stepwise evaluation on examples where the first target token appears in the model's candidate

Metric	AlphaEdit	RCE	Reduction (pp)
First-token hit rate (%)	44.15	27.09	17.06
GFR@FTH (%)	37.88	13.58	24.30
StepR@TP (%)	79.55	30.86	48.69
SufR@FTM (%)	0.00	0.00	0.00

Table 12: Comparison of multi-token target recoverability between AlphaEdit and RCE. RCE consistently reduces first-token exposure and full-string recovery across evaluation settings. StepR@TP and GFR are reported conditionally over examples with a successful first-token candidate-set hit.

Metric	AlphaEdit	HP Def	Max-Ent Def	REVS	RCE
First-token hit rate (%)	44.15	51.84	52.84	1.67	27.09
GFR@FTH (%)	37.88	45.16	46.84	40.00	13.58
StepR@TP (%)	79.55	88.39	89.89	0.00	30.86
SufR@FTM (%)	0.00	0.00	0.00	0.00	0.00

Table 13: Comparison of multi-token target recoverability across AlphaEdit, RCE, and defense baselines.

set. Starting from the ground-truth first token t_1, at each subsequent step $j = 2, \ldots, n$, we check whether the correct token t_j is contained in the candidate set conditioned on the true prefix $(t_1, \ldots, t_{j-1})$. If the current token is hit, we append the ground-truth token and continue to the next step. StepR@TP measures the fraction of first-token-hit examples for which every remaining target token is successfully hit in this stepwise manner.

This metric evaluates whether the correct continuation remains consistently supported by the model, conditioned on both a successful first-token hit and the true history, independent of compounding decoding errors.

- **Suffix recovery after first-token miss. (SufR@FTM)** Finally, we study the failure regime most relevant to our claim. For each multi-token example, we let the model greedily generate a sequence whose length matches that of $\mathbf{y}_{\text{true}}$. We then focus on the cases where the first generated token does *not* match the first token t_1 of $\mathbf{y}_{\text{true}}$. Among those cases, we test whether the remaining generated suffix exactly matches the suffix $(t_2, \ldots, t_n)$ of $\mathbf{y}_{\text{true}}$.

  Formally, let $\mathbf{g} = (g_1, \ldots, g_n)$ denote the generated sequence. We evaluate whether

  $$(g_2, \ldots, g_n) = (t_2, \ldots, t_n) \quad \text{where} \quad g_1 \neq t_1.$$

  This directly tests whether the model can still recover the remainder of the sensitive object after missing the entry token.

We make three observations, summarized in Table 12. First, RCE substantially reduces first-token exposure on multi-token targets, lowering the first-token hit rate from 44.15% to 27.09%. Second, among the cases where the first token is hit, RCE also sharply reduces the recoverability of the complete target string under greedy decoding, from 37.88% to 13.58%. Third, even under the more permissive teacher-forced stepwise evaluation, the full recovery rate drops from 79.55% to 30.86%, indicating that RCE weakens the model's support for the true continuation beyond the first token itself. We further compare RCE with additional defense baselines in Table 13. While the conditional recovery behavior varies across methods, RCE consistently reduces both first-token exposure and continuation recoverability relative to AlphaEdit under greedy and teacher-forced evaluations.

For the SufR@FTM evaluation, we find that once the first token is not generated correctly, the model is generally unable to recover the remaining suffix exactly. This supports the view that the first token functions as a bottleneck for phrase-level leakage: once the autoregressive trajectory deviates at the entry point, the continuation rarely returns to the true sensitive object.

Discussion. These results suggest that first-token suppression has consequences beyond a single-token notion of privacy. Although RCE is optimized only at the first-token level, it also reduces the recoverability of multi-token sensitive objects in practice. This provides empirical support for using first-token rank suppression as a practical proxy for phrase-level protection.

Table 14: Defense performance on the synthetic-PII memorization benchmark. Lower values are better for all metrics.

Defense	HP (%) ↓	PD (%) ↓	IR (%) ↓	RP (%) ↓
AlphaEdit + HP Def.	21.3	28.8	53.8	98.3
AlphaEdit + Max-Ent Def.	21.7	25.0	**0.0**	53.8
AlphaEdit + RCE (Ours)	**10.0**	**12.5**	**0.0**	**23.3**

At the same time, RCE does not explicitly optimize a sequence-level objective over the full sensitive string. Thus, our current method should be understood as a practical first-step intervention rather than a complete solution to phrase-level leakage. Designing explicit sequence-level privacy constraints remains an important direction for future work.

B.11 Synthetic-PII Memorization Evaluation

To evaluate RCE in a setting involving personally identifiable information (PII), we construct a synthetic-PII memorization benchmark using GPT-J-6B. The benchmark contains 240 deletion requests associated with 60 synthetic identities and covers four types of PII: names, email addresses, phone numbers, and physical addresses. All identity information is synthetically generated and does not correspond to real individuals.

For each synthetic identity, the corresponding PII is embedded into naturalistic text passages. To examine the effect of memorization strength, we consider three exposure levels, where each passage containing the target PII is included 1, 5, or 20 times during training. The 240 deletion requests are evenly distributed across the three exposure conditions, resulting in 80 requests per exposure level.

We evaluate RCE together with two baseline defenses applied after AlphaEdit: HP Defense and Max-Entropy Defense. Table 14 reports the results aggregated across all three exposure levels.

As shown in Table 14, RCE achieves the strongest overall defense performance in this controlled setting. Compared with HP Defense, RCE reduces HP from 21.3% to 10.0%, PD from 28.8% to 12.5%, and RP from 98.3% to 23.3%, while completely eliminating IR. Compared with Max-Entropy Defense, RCE maintains the same 0.0% IR while substantially reducing HP, PD, and RP. These results indicate that the effectiveness of RCE extends beyond the primary evaluation setting and remains robust when the target information consists of synthetically memorized PII.

We emphasize that this experiment is intended as a controlled canary-style evaluation of memorization and deletion behavior. Because all identities and PII records are synthetic, the benchmark does not capture the full distributional and contextual complexity of naturally occurring personal information. Therefore, these results should not be interpreted as a substitute for evaluation on real-world personal data.

B.12 Difference from an External Guard Model

We additionally discuss inference-time filtering using an external guard model to better understand how parameter-level defenses and deployment-time safeguards can complement each other. External guard models such as Llama Guard [18] and RCE operate at different layers of the deployment stack. Llama Guard is an auxiliary inference-time component whose protection depends on the downstream system retaining and correctly integrating the guard. In open-weight or self-hosted deployments, such a component can be removed, replaced, or bypassed without modifying the underlying edited model. In contrast, RCE modifies the edited model parameters directly and reduces the residual rank exposure of y_{true} itself. Therefore, RCE provides parameter-level protection that does not rely on an additional guard remaining active at inference time, while external guards can serve as a complementary deployment-time safeguard.

C Qualitative Examples of Rank Leakage and RCE

To complement the aggregate attack-success rates, Table 15 presents two illustrative CounterFact examples under the single-edit setting with GPT-J-6B and AlphaEdit. We use the same RCE

configuration as in the main experiments ($\tau = 5000$ and $m = 5$) and evaluate RP with a candidate budget of $B = 20$. For each state, we report the greedy next-token output, the five highest-ranked tokens and their pre-softmax logits, and the rank of the original sensitive token y_{true}.

Prompt / edit	Original model	AlphaEdit	AlphaEdit + RCE
Case 1 *The mother tongue of Danielle Darrieux is* French → English	**Greedy** French TOP-5 LOGITS French (17.77) • the (15.09) • not (15.07) • English (14.25) • a (13.86) **Rank of y_{true}** 1	**Greedy** English TOP-5 LOGITS English (21.03) • the (16.09) • British (15.58) • english (15.32) • not (14.37) **Rank of y_{true}** 12 **RP@20** EXPOSED	**Greedy** English TOP-5 LOGITS English (25.26) • the (19.55) • American (18.11) • english (18.08) • not (17.07) **Rank of y_{true}** 50,297 **RP@20** BLOCKED
Case 2 *The official religion of Jahangir is* Islam → Judaism	**Greedy** Islam TOP-5 LOGITS Islam (15.38) • the (14.54) • Hindu (13.84) • a (13.20) • that (12.95) **Rank of y_{true}** 1	**Greedy** Judaism TOP-5 LOGITS Judaism (24.44) • the (16.54) • Jewish (16.48) • Zionism (16.48) • Islam (15.76) **Rank of y_{true}** 5 **RP@20** EXPOSED	**Greedy** Judaism TOP-5 LOGITS Judaism (21.53) • the (15.07) • Jewish (14.63) • a (13.74) • that (12.80) **Rank of y_{true}** 5,188 **RP@20** BLOCKED

Table 15: Qualitative examples of rank leakage and its mitigation. Tokens are ordered by next-token logit, with logits in parentheses; display-leading whitespace is omitted. The original sensitive token is red and underlined, whereas the sanitized token is green and bold. EXPOSED indicates that y_{true} remains within the RP@20 candidate set; BLOCKED indicates that it falls outside this set.

These examples show why rewrite fidelity alone does not establish privacy. AlphaEdit changes the greedy answer to y_{new}, yet y_{true} remains at ranks 12 and 5, respectively, and is therefore exposed by a single RP query. RCE retains the sanitized answer at rank 1 while moving the sensitive token to ranks 50,297 and 5,188, outside the $B = 20$ attack budget. As qualitative illustrations, these cases complement—rather than replace—the aggregate evaluation of candidate exposure.

D Limitations

As in prior work [4, 27], RCE aims to reduce practical extractability under budget-constrained adversaries rather than to guarantee information-theoretic deletion. An adversary with a larger candidate-verification budget and sufficiently broad top-k access may still recover sensitive information, although verifying more candidates incurs additional attack costs. Moreover, RankPeek assumes access to top-k candidate token rankings, which are not available through all model APIs. When only a single output is available per query, RankPeek cannot be applied directly; we instead evaluate a separate repeated-query sampling attack in Appendix B.8. Another limitation is that RCE directly constrains only the first token of a sensitive answer, rather than optimizing protection over the entire sequence. Finally, although RCE substantially reduces extraction success across different models, its impact on edit fidelity and model utility varies across architectures.

E Broader impacts

This work aims to improve privacy protection in edited large language models by reducing the risk that sensitive facts remain recoverable after knowledge editing. The proposed defense may benefit applications where model providers need to remove or sanitize private, outdated, or sensitive information while preserving model utility. At the same time, our work highlights that editing the top-1 output alone may be insufficient for privacy protection, which can help practitioners better evaluate the residual leakage risks of edited models.

However, the method also has potential negative impacts. First, a defense that reduces extraction success may create a false sense of security if deployed without comprehensive privacy evaluation, since adaptive or stronger attacks may still reveal sensitive information. Second, techniques for

concealing model knowledge could be misused to hide information in non-transparent ways, such as suppressing facts for censorship, evasion of auditing, or manipulation of model behavior. Therefore, RCE should be used together with careful privacy auditing, transparent documentation of editing objectives, and evaluation against diverse extraction attacks before deployment in sensitive or high-stakes settings.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the proposed method, experimental setting, and main empirical findings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the proposed method in Appendix D

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the datasets, experimental setup, implementation details, and evaluation protocol required to reproduce the main results. Additional hyperparameters and training details are provided in Appendix A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is available at `https://github.com/JasonH0810/RCE`

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the experimental settings, including datasets, train/validation/test splits, hyperparameters, optimizer, training procedure, and evaluation metrics in Section 5, Section 6, Appendix A and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We assess the stability of our results through five independent runs with different random seeds and five independently sampled evaluation subsets. The repeated runs exhibit negligible variability, while the subset-based evaluation reports mean and standard deviation across five subsets of 500 CounterFact instances. Detailed protocols and results are provided in Appendix A.5 and Table 7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the compute resources used for the experiments, including GPU type, memory, running time per experiment in Appendix A.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms to it. The work does not involve unethical data collection, privacy violations, or other procedures that would violate the code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We discuss broader impacts in Appendix E. The work may have positive societal impacts by improving privacy protection in edited large language models and reducing the risk of extracting sensitive facts, while potential negative impacts include a false sense of privacy, misuse for non-transparent knowledge suppression, and the need for careful auditing before deployment.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This question is not applicable because the paper does not release new high-risk models or datasets. The proposed method is a defense for reducing privacy leakage in edited language models, and the experiments are conducted on existing models and benchmark datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our experiments use existing language models, datasets, editing algorithms, and extraction attack baselines. We cite the original papers or repositories for all models, datasets, and methods used in our evaluation.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: This question is not applicable at submission time because the paper does not release new datasets, model checkpoints, or other assets. We plan to release the RCE implementation and evaluation scripts upon acceptance with appropriate documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This question is not applicable because the paper does not involve crowdsourcing, user studies, or research with human subjects. All experiments are conducted on existing models and benchmark datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This question is not applicable because the paper does not involve crowdsourcing or research with human subjects. Therefore, IRB approval or equivalent review is not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: We use GPT-4.1 to generate paraphrased prompts for the Input Rephrasing attack in our experimental evaluation. This usage is limited to constructing attack prompts under explicit semantic-preservation constraints and does not constitute part of the proposed RCE optimization method. The details, including the paraphrase-generation constraints and prompt templates, are described in Appendix A.4.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.