KEMU: Keyword-aware Embedding Modification for Efficient Unlearning

Anonymous ACL submission

Abstract

Current unlearning methods depend on broad parameter updates and rely on retain set supervision or reference model constraints to prevent catastrophic forgetting, incurring substantial computational overhead. Through empirical analysis, we identify the embedding layer as a representational bottleneck where targeted modifications to a small number of token embeddings can globally reshape downstream representations. Based on this insight, we propose KEMU, a framework that achieves unlearning by editing only the embeddings of a minimal set of forget-specific tokens. KEMU employs an importance scoring mechanism to rapidly pinpoint critical tokens with a bounded regularization strategy to restrict semantic drift. Formal bounds on embedding deviation and retained knowledge degradation further support the idea We validate KEMU on both encoder-only and decoder-only language model architectures. KEMU achieves competitive performance on both classification and generative tasks, reducing parameter updates by 2 to 5 orders of magnitude compared to baselines while effectively preserving general utility. Codes are available at https://anonymous.4open.science/r/KEMU-6011/.

1 Introduction

As large language models are deployed in privacy-sensitive applications, selectively removing specific knowledge from trained models has become a critical requirement (Union, 2016; Bourtoule et al., 2021). Early methods such as gradient ascent (GA) (Yao et al., 2024) optimize directly against the forget objective without external constraint, leaving the model vulnerable to unbounded parameter drift. Although computationally lightweight, they suffer catastrophic forgetting of retained knowledge as the forget set grows (Maini et al., 2024).

Recent works restrict the update through retain-set supervision (Maini et al., 2024; Zhang et al., 2024; Fan et al., 2026; Entesari et al., 2025) or frozen reference models (Li et al., 2024; Dong et al., 2025) to avoid over forgetting. However, both retain-set and reference models introduce heavy memory constraints and inflate training latency. Such high burden derived from the fact that broad parameter updates entangle forget-relevant changes with retained knowledge, making it challenging to decouple them without extensive regularization.

We study whether a single, small model component can achieve effective unlearning with minimal collateral damage. Since token identities enter the computation graph exclusively through the embedding layer regardless of architecture, this makes the embedding layer a universal intervention point. To verify that it functions as a high-leverage bottleneck, we use BERT as a diagnostic probe for a layer-wise sensitivity analysis. We find that the embedding layer acts as a high-leverage bottleneck: while Unconstrained updates on embedding layer cause catastrophe forgetting due to representation collapse (Figure 1), sparse edits targeting a minimal fraction of forget-specific tokens are sufficient to approach retrain-level performance (Figure 2).

These results indicate that forget-specific and retain-relevant token embeddings occupy largely separable subspaces. If editing operations are strictly confined to the former, the latter remains approximately intact. This modularity eliminates the need for costly retain-set supervision or heavy reference models during gradient updates. Importantly, because the embedding layer serves as the identical token-to-representation role in all Transformers, this insight applies universally across architectures. This foundational property motivates us to design a unified, highly efficient unlearning framework applicable to both encoder and decoder models.

Leveraging this functional invariance of the embedding layer and the separability of the embedding space, we propose KEMU (Keyword-aware Embedding Modification for Efficient Unlearning).

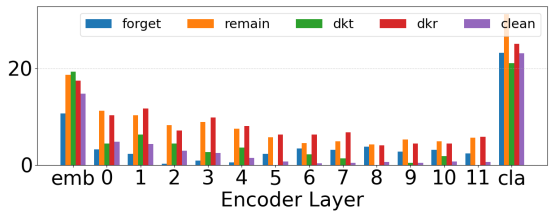

Figure 1: Layer-wise unlearning performance (IMDb forget-1%). Each layer was individually unfrozen and fine-tuned. Bars show the absolute difference from Retrain performance. Unconstrained updates to the embedding layer cause catastrophic forgetting.

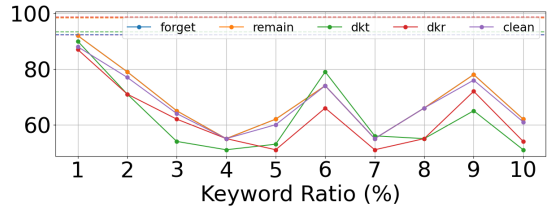

Figure 2: Different keyword ratio performance (IMDb forget-1%). Horizontal line marks retrain performance. A keyword ratio of 1% approaches retrain-level forgetting, exceeding this threshold degrades utility.

Achieving unlearning by editing the embeddings of a minimal set of forget-specific tokens, KEMU has no need for expensive reference models or retain-set supervision. Specifically, tokens are identified via an importance scoring mechanism, and their embeddings are updated via a confusion-inducing objective. This localized intervention minimizes parameter updates while preserving model behavior on retained data. To further constrain side effects, we introduce a bounded regularization strategy that limits embedding drift in the representation space. Theoretical analysis establishes a worst-case bound on retained knowledge degradation under localized embedding edits (Section 2.5). Our contributions are as follows:

We propose KEMU, a resource-efficient unlearning framework that edits only the embeddings of a minimal set of forget-specific tokens. Recognizing the embedding layer as a high-leverage representational bottleneck, KEMU confines all updates to this layer, removing the need for reference models or retain-set supervision during training.

We instantiate KEMU for both encoder and decoder architectures. For encoders, tokens are scored by Fisher information and updated toward random class labels. For decoders, tokens are ranked by forget-to-retain frequency ratios and steered toward "I don't know" responses. A bounded regularization strategy limits embedding drift, supported by a worst-case theoretical bound on retained knowledge degradation.

We evaluate KEMU across both encoder-only and decoder-only models, demonstrating that the separable subspace property is a fundamental characteristic of Transformer embeddings rather than an artifact of a specific architecture. Our empirical results show that KEMU matches or exceeds the unlearning efficacy of existing methods while preserving comparable utility, and modifies orders of magnitude fewer parameters than all baselines.

2 Methodology

2.1 Overview

Existing unlearning methods each address the forgetting-utility trade-off from a different angle, but share a common cost. GA (Yao et al., 2024) applies gradient ascent directly on the forget objective, which is computationally cheap but causes catastrophic forgetting as updates propagate indiscriminately across all parameters. GradDiff (Maini et al., 2024) and SimNPO (Fan et al., 2026) address this by penalizing deviations from a retain set during training, but this requires continuous access to retain data and scales poorly with retain-set size. NPO (Zhang et al., 2024) and UNDIAL (Dong et al., 2025) each rely on a frozen reference model, doubling the memory footprint at training time. RMU (Li et al., 2024) steers intermediate hidden states toward random vectors, but the perturbation is applied to shared representations and causes collateral damage to retained knowledge. PDU (Entesari et al., 2025) achieves stronger forgetting through primal-dual optimization, but modifies billions of parameters per run and requires retain-set supervision during training. The main difficulty is that all these methods operate over the full parameter space or shared intermediate representations, making it hard to confine updates to forget-relevant computations without external regularization. KEMU takes a different approach. By restricting updates to the embedding layer and further limiting edits to a small set of forget-specific tokens, KEMU confines the modification to parameters whose influence on the retain set is negligible by construction. This structural locality eliminates the need for retain-set supervision or reference models during training, while modifying orders of magni-

tude fewer parameters than all compared methods.

Specifically, KEMU achieves machine unlearning through targeted modification of a small set of token embeddings identified as most influential for the forget set $\mathcal{D}_f$. The framework consists of two components: (1) **Keyword Selection**: identifies forget-specific tokens via importance scoring, and (2) **Selective Representation Modification**: edits only the selected token embeddings via gradient-guided updates with norm-preserving regularization. While the core rationale is architecture-agnostic, the implementation differs between encoder-only and decoder-only models.

2.2 Problem Formulation

Let $\mathcal{D} = (x_i, y_i)_{i=1}^N$ be the full training set and $\mathcal{D}_f \subset \mathcal{D}$ the forget set, with $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. A pre-trained model $f(\cdot;\theta_p)$ is fine-tuned on $\mathcal{D}$ to obtain $f(\cdot;\theta)$. The goal of unlearning is to produce $\hat{\theta}$ such that $f(\cdot;\hat{\theta})$ behaves as if $\mathcal{D}_f$ had never been included in training, while preserving performance on $\mathcal{D}_r$. KEMU achieves this by modifying only the embedding rows indexed by a small keyword set $\mathcal{K}$: $\hat{\theta}^{e^k} \neq \theta^{e^k}$ only for $k \in \mathcal{K}$, and $\hat{\theta}^{(\neg e)} = \theta^{(\neg e)}$ for all non-embedding parameters.

2.3 Keyword Selection

The keyword selection module aims to identify tokens that are most semantically relevant to the samples to be forgotten. We first present a general form of importance scoring, then describe architecture-specific instantiations including **Fisher Scoring** and **Specificity Scoring**.

2.3.1 General Formulation.

The proposed keyword selection module assigns an importance score $s(t)$ to each token t in the vocabulary $\mathcal{V}$, reflecting its relevance to the information to be forgotten. The keyword set $\mathcal{K}$ is then formed by selecting the top-K influential tokens:

$$\mathcal{K} = \underset{|\mathcal{K}|=K}{\arg\max} \sum_{t \in \mathcal{K}} s(t). \tag{1}$$

A token filter $\mathcal{F}$ is applied prior to scoring to exclude uninformative tokens. All the special tokens, punctuation, and stop-words are eliminated.

2.3.2 Fisher Scoring

For encoder-only models, the responsiveness of the model's predictions to perturbations of a given input token reflects its semantic relevance. We treat this responsiveness as the token's importance and quantify it via Fisher scoring. For each token t in an input sequence, we first compute the gradient of the loss with respect to its embedding $g_t = \nabla_{e_t}\mathcal{L}$, then form an influence vector by reweighting the gradient with the inverse Fisher information:

$$s(t) = \left\| g_t \odot \frac{1}{g_t^2 + \lambda} \right\|_2, \tag{2}$$

where $g_t^2 + \lambda$ provides a damped diagonal approximation of the Fisher Information Matrix, with damping constant λ ensuring numerical stability. This score reflects the model's sensitivity to perturbations of token t.

Subword tokens are subsequently grouped by detokenization (e.g., `play` + `##ing` $\rightarrow$ `playing`) and scored by their constituent average. To select instance-level keywords, we aggregate consecutive n-token spans by averaging their scores and pick the top-ranked n-grams per sample, yielding interpretable natural language keywords. More details can be found in the Appendix B, Algorithm 1.

2.3.3 Specificity Scoring

Unlike encoder-only models, computing Fisher scores incurs considerable computational overhead for decoder-only models due to their substantially larger parameter counts. To preserve the efficiency of KEMU, we instead compute a specificity score based on the relative frequency of each token in $\mathcal{D}_f$ versus $\mathcal{D}_r$:

$$s(t) = \frac{f_t/F}{r_t/R + \epsilon} \cdot \phi(f_t), \tag{3}$$

where f_t and r_t denote the token counts in $\mathcal{D}_f$ and $\mathcal{D}_r$ respectively, $F = \sum_{t'} f_{t'}$ and $R = \sum_{t'} r_{t'}$ are the respective total counts, and ϵ is a smoothing constant. Let $\tilde{s}(t) = \frac{f_t/F}{r_t/R+\epsilon}$ denote the raw specificity before frequency re-weighting, so that $s(t) = \tilde{s}(t) \cdot \phi(f_t)$. The frequency penalty $\phi(f_t)$ uses a low-frequency threshold τ_{lf} to suppress tokens with low occurrence counts:

$$\phi(f_t) = \min\left(1, \frac{f_t}{\tau_{\text{lf}}}\right), \tag{4}$$

where τ_{lf} is the threshold below which scores are linearly suppressed.

This formulation requires only offline token frequency statistics over $\mathcal{D}_r$. Access to $\mathcal{D}_r$ is not required during training. Tokens with high specificity scores are by construction rare in $\mathcal{D}_r$, providing implicit retain performance protection. Editing these

embeddings has negligible impact on retain-set representations, eliminating the need for retain data during the gradient update process. More details can be found in Appendix B, Algorithm 2.

2.4 Selective Representation Modification

To alter the semantic representation of samples in $\mathcal{D}_f$, we adopt a confused label strategy. For each sample $(x_i, y_i) \in \mathcal{D}_f$, we replace the original label $= y_i$ with a confusing alternative $\hat{y}_i$, weakening the association between input texts and their ground-truth labels. The specific construction of $\hat{y}_i$ is architecture and task dependent, confused label and IDK response are generally used as targets for classification and generation tasks. Detailed settings can be found in Appendix B.

We freeze all model parameters except the embedding layer θ^e, and fine-tune only the parameters $\theta^{e^{\mathcal{K}}}$ associated with $\mathcal{K}$ extracted from $\mathcal{D}_f$. The unlearning objective is:

$$\mathcal{L}_{\text{CE}}(x, \hat{y};\ \theta^{e^{\mathcal{K}}}) = -\sum_{(x,\hat{y})\in\mathcal{D}_f} \log f(\hat{y} \mid x;\ \theta^{e^{\mathcal{K}}}). \tag{5}$$

The keyword embeddings are updated via gradient descent on the confused label loss $\mathcal{L}_{\text{CE}}$:

$$\hat{\theta}^{e^{\mathcal{K}}} \leftarrow \theta^{e^{\mathcal{K}}} - \eta \cdot \frac{\nabla_{\theta^{e^{\mathcal{K}}}} \mathcal{L}_{\text{CE}}}{\|\nabla_{\theta^{e^{\mathcal{K}}}} \mathcal{L}_{\text{CE}}\|_2}, \tag{6}$$

where η is the step size controlling the update magnitude. To prevent embeddings from drifting to degenerate magnitudes, we apply norm-preserving rescaling after each update:

$$\hat{\theta}^{e^k} \leftarrow \hat{\theta}^{e^k} \cdot \frac{\|\theta_0^{e^k}\|_2}{\|\hat{\theta}^{e^k}\|_2}, \quad \forall k \in \mathcal{K}, \tag{7}$$

where $\theta_0^{e^k}$ denotes the embedding of the pre-unlearned model, serving as the reference for magnitude preservation. This operation preserves the original vector magnitude, preventing embeddings from over-drifting in representation space.

For privacy-sensitive encoder-only scenarios where original forget samples cannot be exposed, KEMU additionally provides a keyword-aware only mode that constructs pseudo-samples from extracted keywords and an irrelevant corpus, enabling unlearning without access to the original forget data. More details are provided in Appendix D.1.

2.5 Theoretical Analysis

We provide theoretical justification for three design choices. First, the confused label mechanism guarantees that maximizing $P(\hat{y}|x)$ suppresses $P(y|x)$, directly erasing the target knowledge. Second, the gradient-direction analysis establishes that the ℓ_2-normalized negative gradient provides the steepest per-unit-norm reduction of the loss, giving the optimal per-token edit direction. Third, Theorem 1 bounds the expected output perturbation on $\mathcal{D}_r$, confirming that norm-preserving regularization keeps collateral damage under control. Since token identities reach the computation graph exclusively through the embedding layer ($f = g \circ \mathbf{E}$ in Theorem 1), editing only keyword embeddings selectively affects forget-specific inputs while structurally isolating retain data.

confused label Mechanism. KEMU minimizes $\mathcal{L}_{\text{CE}}(x, \hat{y})$ with respect to a confused label $\hat{y}$. The softmax enforces $\sum_y P(y|x) = 1$, so maximizing $P(\hat{y}|x)$ necessarily suppresses $P(y|x)$, thereby erasing the target knowledge. For decoder-only models, the per-position softmax at each decoding step operates identically to the encoder case, suppressing the factual token probability in favor of the IDK response at each position.

Gradient Direction. Let $e_t \in \mathbb{R}^D$ denote the embedding vector of token t. KEMU updates embeddings along the negative gradient of the confused label loss:

$$\hat{\mathbf{d}}_t = -\frac{\nabla_{e_t} \mathcal{L}_{\text{CE}}(x, \hat{y})}{\|\nabla_{e_t} \mathcal{L}_{\text{CE}}(x, \hat{y})\|_2}, \tag{8}$$

gives the steepest per-unit-norm reduction of $\mathcal{L}_{\text{CE}}$. By the softmax constraint, this simultaneously suppresses $P(y|x)$. The update is applied only to tokens identified by Fisher scoring (encoders) or the frequency-ratio proxy (decoders). ℓ_2-normalized gradients ensure uniform step magnitude across selected tokens.

Preservation of Retained Performance. KEMU limits collateral damage through specificity-based keyword selection and norm-preserving rescaling. After each gradient update (Eq. 6), we rescale the modified embedding to match its original ℓ_2 norm (Eq. 7). This prevents embeddings from drifting to degenerate magnitudes during multi-step editing. The norm constraint bounds the perturbation by $|\Delta e_t|_2 \leq 2|e_t^{(0)}|_2$, which feeds into the retain-side analysis of Theorem 1.

Theorem 1 (Retain Performance Bound)**.** *Decompose $f = g \circ \mathbf{E}$ where $\mathbf{E} \in \mathbb{R}^{V\times D}$ is the embedding layer and g is the subsequent Transformer stack. Editing a small keyword set $\mathcal{K}$ can therefore disrupt forget-relevant computations. Let L_E*

be the Lipschitz constant of g taken over the ℓ_2-ball of radius $2\|e^{(0)}\|_{\max}$ centered at $\mathbf{E}^{(0)}$. Let f_t, r_t be token frequencies in $\mathcal{D}_f$, $\mathcal{D}_r$, with totals $F = \sum_t f_t$, $R = \sum_t r_t$. Let $\mathcal{K}$ satisfy $\tilde{s}(t) \geq \tau_{spec}$ for all $t \in \mathcal{K}$, where $\tilde{s}(t) = \frac{f_t/F}{r_t/R+\epsilon}$ denotes the raw specificity (Eq. 3 without the frequency penalty). Here, τ_{spec} controls how skewed the selected keywords are toward the forget set relative to the retain set. A larger τ_{spec} means the chosen keywords appear more rarely in the retain data, so editing them causes less collateral impact on retained performance. This implies $r_t/R \leq f_t/(\tau_{spec}F)$ for $\epsilon \ll \tau_{spec}$. After a norm-preserving update with $\|\Delta e_t\|_2 \leq 2\|e_t^{(0)}\|_2$, the expected output change on $\mathcal{D}_r$ satisfies:

$$\mathbb{E}_{x\sim\mathcal{D}_r}[\|\Delta f(x)\|] \leq \frac{2L_E \cdot |\mathcal{K}| \cdot \|e^{(0)}\|_{\max}}{\tau_{spec}} \cdot \frac{f_{\max}}{F}, \quad (9)$$

where $\Delta f(x) = f(x;\hat{\theta}) - f(x;\theta)$ denotes the change in model output, $\|e^{(0)}\|_{\max} = \max_t \|e_t^{(0)}\|_2$ and $f_{\max} = \max_{t\in\mathcal{K}} f_t$.

Theorem 1 bounds the expected output perturbation on $\mathcal{D}_r$; it serves as a controllability check confirming that keyword-targeted edits do not induce unbounded downstream drift. A complete proof is provided in Appendix G. Empirical retain-side performance is evaluated in Sec. 3.2. The bound in Theorem 1 can be calibrated empirically using gradient norms logged during training; this analysis, confirming that the bound is quantitatively consistent with the observed retain performance, is provided in Appendix G.1.

3 Experiments

3.1 Experimental Setup

3.1.1 Datasets and Tasks

Encoder-only. We evaluate on two standard sentiment classification benchmarks: IMDb (Maas et al., 2011) and SST-2 (Socher et al., 2013). By default, we randomly designate $\beta = 1\%$ of the training instances as $\mathcal{D}_f$ and treat the remainder as $\mathcal{D}_r$. To assess semantic erasure, we also construct semantically closest subsets ($\mathcal{D}_{kr}$ and $\mathcal{D}_{kt}$) by applying k-means clustering to the retain and test sets, respectively. Following KGA (Wang et al., 2023), we treat proximity to the Retrain model as the gold standard for encoder-side evaluation. **Decoder-only.** We use two complementary benchmarks. (i) TOFU (Maini et al., 2024) comprises fictitious author biographies specifically designed to evaluate fine-grained factual unlearning in LLMs. We evaluate across three subsets of increasing forget scale: forget01 (1%), forget05 (5%), and forget10 (10%), each paired with a retain split covering the remaining authors. (ii) MUSE (News domain) (Shi et al., 2024) evaluates unlearning of long-form contiguous knowledge from BBC news articles, where the forget and retain corpora are disjoint sets of documents. MUSE differs from TOFU in evaluating unlearning on continuous news articles rather than discrete Q&A pairs, with separate metrics for verbatim memorization and retained knowledge.

3.1.2 Baselines

For encoder-only tasks, we compare KEMU with gold standard retrain model (Ret), fine-tuning (Fin), gradient ascent (GA), and KGA (Wang et al., 2023). For decoder-only tasks, we compare against seven classic or the latest baseline, including: Gradient Ascent (GA (Yao et al., 2024), GradDiff (Maini et al., 2024), NPO (Zhang et al., 2024), SimNPO (Fan et al., 2026)), RMU (Li et al., 2024), PDU (Entesari et al., 2025), and UNDIAL (Dong et al., 2025). Detailed settings for all baselines are provided in Appendix C.3.

3.1.3 Evaluation Metrics

Encoder-only. To quantify unlearning efficacy and utility preservation, we report inference Accuracy and F1-score across $\mathcal{D}_f$, $\mathcal{D}_r$, and the test set $\mathcal{D}_t$. Furthermore, we evaluate performance on the semantically similar subsets ($\mathcal{D}_{kr}$ and $\mathcal{D}_{kt}$) to measure the extent of semantic knowledge erasure. Finally, we assess privacy protection capabilities using Membership Inference Attacks (MIA), reporting the change in Attack Success Rate (ΔASR $\downarrow$) and the Area Under the Curve deviation from random guessing (ΔAUC $\downarrow$). Across all encoder baselines, we additionally report $\Delta P \downarrow$ as the number of parameters actually modified (in billions) to highlight update efficiency.

Decoder-only. Following standard protocols (Maini et al., 2024; Dong et al., 2025), we report four complementary metrics on TOFU. To assess unlearning efficacy, we measure the Forget Truth Ratio (FTR) $\uparrow$ (the proportion of responses successfully redirected to "I don't know" style answers) and Forget Q&A ROUGE (F-RG) $\downarrow$ (residual factual overlap). To evaluate utility preservation, we report Model Utility (MU) $\uparrow$ (a

Table 1: Comparison of unlearning methods on IMDb and SST-2 across three forget splits.

Dataset	Metric	Acc						F1					
		Ori	Ret	Fin	GA	KGA	**KEMU**	Ori	Ret	Fin	GA	KGA	**KEMU**
IMDb	$\mathcal{D}_f$	98.65	91.93	**90.13**	98.21	96.86	96.41	98.65	91.93	**90.09**	98.20	96.85	96.40
	$\mathcal{D}_r$	99.12	98.25	98.13	98.97	99.89	**98.28**	99.12	98.25	98.13	98.97	99.89	**98.28**
	$\mathcal{D}_{kt}$	91.48	93.72	**93.27**	91.48	**93.27**	**93.27**	91.46	93.72	93.26	91.46	**93.27**	93.26
	$\mathcal{D}_{kr}$	99.55	97.76	96.86	99.55	100.00	**96.41**	99.55	97.76	96.86	99.55	**100.00**	97.76
	$\mathcal{D}_t$	92.28	92.48	90.60	92.32	**92.84**	91.88	92.27	92.47	90.59	**92.31**	92.84	91.87
SST-2	$\mathcal{D}_f$	96.20	93.06	**93.39**	95.87	99.83	95.21	96.14	92.92	**93.30**	95.81	99.83	95.14
	$\mathcal{D}_r$	97.20	97.06	96.76	**97.13**	99.96	98.86	97.16	97.02	96.72	**97.10**	99.96	98.84
	$\mathcal{D}_{kt}$	97.19	97.85	97.52	97.02	58.02	**98.51**	96.22	97.06	96.67	96.01	**57.43**	97.97
	$\mathcal{D}_{kr}$	97.69	97.19	95.37	**97.69**	100.00	98.84	97.66	97.15	95.32	**97.66**	100.00	98.83
	$\mathcal{D}_t$	94.46	94.37	93.94	**94.49**	94.52	94.85	94.42	94.30	93.89	**94.45**	**94.45**	94.80
	$\Delta P\downarrow$	—	—	1.09e-01	1.09e-01	9.86e-02	**1.26e-03**						

composite score of general language capabilities on $\mathcal{D}_r$) and Retain Q&A ROUGE (R-RG) $\uparrow$. On MUSE, following (Shi et al., 2024), we report Forget Knowledge (FK) $\downarrow$, Forget Verbatim (FV) $\downarrow$, and PrivLeak (PL) to quantify residual memorization and privacy leakage, and Retain Knowledge (RK) $\uparrow$ to measure utility on the retain corpus. Across both benchmarks, we additionally report $\Delta P \downarrow$ as the number of parameters actually modified (in billions) to highlight update efficiency. Since baseline methods train with an FP32 optimizer but store weights in BF16, updates below the quantization threshold are discarded on write-back, causing the reported ΔP of full-model methods (e.g., GradAscent, GradDiff) to fall below the total parameter count.

3.1.4 Implementation Details

KEMU is designed to unlearn all downstream knowledge acquired through fine-tuning a pre-trained model, rather than to unlearn the pre-trained knowledge itself. Consequently, all experiments were conducted based on the fine-tuned pre-trained model. For encoder-only experiments, we implement all methods based on the bert-base-uncased model. KGA was fine-tuned by its default settings to maintain a fair and robust performance comparison. For decoder-only experiments, we deploy all methods with open-unlearning framework on the Llama-2-7b-chat-hf model. For Fisher scoring (Eq. 2), we set the damping constant $\lambda = 10^{-3}$. For specificity scoring (Eq. 3), we use $\epsilon = 10^{-4}$ and $\tau_{\text{lf}} = 3$. Qualitative examples of extracted keywords are provided in Appendix F. Comprehensive hyperparameter and environmental settings are detailed in Appendix C.

3.2 Main Results

3.2.1 Encoder-only Results

In all result tables, **bold** and underline denote the best and second-best results among reported methods, respectively. Table 1 shows the comparison results on sentiment analysis task. Here, we set 3 cluster centers and 1 nearest neighbor sample to observe the performance of $\mathcal{D}_{kr}$ and $\mathcal{D}_{kt}$.

Fisher scoring isolates distinct embedding subspaces. KEMU achieves competitive or superior performance in most settings compared to baselines, which proves that the keywords we selected can isolate the representations of forget and retain set in the embedding space. KEMU attains this result with 1.26×10^{-3} B parameter updates, two orders of magnitude below full-model methods, and a single training epoch, since keyword selection restricts all modifications to a small embedding subset. Notably, although KGA reports a net ΔP of 9.86×10^{-2} B, each of its two intermediate stages performs full-model fine-tuning, so the modified parameter far exceeds the report.

Embedding-layer edits provide privacy and context-dependent control. Under membership inference attacks, KEMU surpasses the retrain baseline in indistinguishability because most parameters remain untouched, keeping the model's overall behavior closely aligned with its pre-unlearning state. KEMU_{KO} effectively reduces retention of forgotten knowledge but shows slightly decreased retain-set accuracy due to over-removal on semantically close subsets. Details can be found in Appendix D. The keyword-only (KO) mode uses the same confused label objective and effectively reduces retained knowledge, but KEMU_{KO} shows slightly decreased $\mathcal{D}_r$ accuracy and over-removal on semantically close subsets ($\mathcal{D}_{kt}$, $\mathcal{D}_{kr}$), since pseudo-samples built from keywords omit non-

Table 2: Comparison of unlearning methods on TOFU across three forget splits.

Method	forget01 (1%)					forget05 (5%)					forget10 (10%)				
	FTR↑	F-RG↓	R-RG↑	MU↑	ΔP↓	FTR↑	F-RG↓	R-RG↑	MU↑	ΔP↓	FTR↑	F-RG↓	R-RG↑	MU↑	ΔP↓
Retrain	0.684	0.397	0.618	0.703	-	0.670	0.395	0.613	0.661	-	0.675	0.397	0.618	0.660	-
GA	0.574	0.425	0.707	0.588	1.8e+00	0.650	0.243	0.353	0.537	2.2e+00	0.575	0.557	0.635	0.627	1.3e+00
GradDiff	0.571	0.422	0.556	0.597	3.3e+00	0.630	0.399	0.433	0.582	4.5e+00	0.693	0.342	0.368	0.526	5.5e+00
NPO	0.672	0.351	0.477	0.568	4.0e+00	0.719	0.241	0.387	0.565	4.2e+00	0.720	0.222	0.398	0.563	4.7e+00
PDU	0.742	0.158	0.560	0.608	2.1e+00	**0.806**	**0.133**	0.464	0.578	3.1e+00	**0.879**	**0.077**	0.646	0.614	4.0e+00
RMU	0.529	0.905	**0.870**	0.615	1.2e-01	0.524	0.817	0.837	0.607	1.2e-01	0.520	0.823	0.835	0.618	1.3e-01
SimNPO	0.602	0.400	0.575	0.602	3.3e+00	0.657	0.335	0.478	0.581	3.6e+00	0.678	0.347	0.439	0.588	4.3e+00
UNDIAL	0.646	0.332	0.767	**0.645**	1.9e+00	0.659	0.281	0.601	**0.643**	3.5e+00	0.658	0.278	0.632	**0.642**	3.9e+00
KEMU	**0.799**	**0.041**	0.865	0.616	**4.9e-05**	0.734	0.233	**0.865**	0.615	**1.7e-04**	0.725	0.187	**0.867**	0.606	**2.1e-04**

keyword context such as negation markers and intensifiers that full-sample KEMU preserves. Details can be found in Appendix D

3.2.2 Decoder-only Results

TOFU benchmark Table 2 summarizes the comparison results on the TOFU benchmark across the 1%, 5%, and 10% forget splits.

Targeted embedding edits achieve effective unlearning with minimal parameter cost. KEMU maintains an FTR above 0.69 and R-RG near 0.86 across all three splits while modifying four to five orders of magnitude fewer parameters than gradient-based baselines. This efficiency follows directly from confining updates to forget-specific keyword embeddings, which carry the bulk of forget-relevant information while leaving retain-relevant parameters untouched.

Retain regularization trades unlearning depth for utility, while embedding edits avoid this trade-off. GradDiff and SimNPO maintain stable utility by penalizing retain deviations, but their F-RG scores remain above 0.33 despite modifying over 3.3×10^9 parameters, indicating that retain constraints suppress forgetting. KEMU achieves lower F-RG without any retain-set supervision, because specificity-based keyword selection structurally isolates forget-relevant embeddings from retain-relevant ones, making external regularization unnecessary.

Full-model optimization degrades utility as the forget set grows. GA and PDU show increasing utility degradation on `forget05` and `forget10`, with MU dropping below baseline levels. Broad parameter updates entangle forget and retain knowledge in shared representations, and this entanglement becomes more severe as the forget set expands. KEMU's MU remains stable at 0.61 to 0.62 across all splits, consistent with the retain performance bound in Theorem 1. In terms of computational cost, Table 3 shows that KEMU completes unlearning in under 0.01 GPU-hours with 16.1 GB of peak memory on forget01, substantially lower than all baselines. Full resource statistics for `forget05` and `forget10` are provided in Appendix E.5.

Table 3: Resource cost on TOFU.

Method	Time (h)↓	Mem (GB)↓	GPU Hrs↓
GA	0.04	23.0	0.08
GradDiff	0.03	23.0	0.07
NPO	0.03	23.1	0.08
PDU	0.03	23.0	0.07
RMU	0.01	23.0	0.03
SimNPO	0.03	23.0	0.07
UNDIAL	0.05	23.1	0.18
KEMU	**0.01**	**16.1**	**0.02**

Table 4: Comparison of unlearning methods on MUSE-News benchmark.

Method	FK↓	FV↓	PL	RK↑	ΔP↓
GradDiff	0.526	0.420	-95.90	0.461	5.4e+00
NPO	0.581	0.370	-84.88	0.515	5.7e+00
RMU	0.221	0.167	-51.72	0.229	1.3e-01
SimNPO	0.504	0.365	-96.52	0.469	5.4e+00
UNDIAL	0.446	0.176	-98.60	0.372	5.1e+00
PDU	0.438	**0.043**	92.48	0.388	4.9e+00
KEMU	**0.009**	0.054	**26.74**	**0.527**	**6.6e-05**

3.2.3 MUSE Results

MUSE benchmark **Keyword-targeted edits achieve strong forget performance without damaging retain knowledge.** KEMU achieves FK $= 0.009$ and FV $= 0.054$, outperforming RMU by $11.3\times$ on FK and $2.4\times$ on FV, while attaining the highest retain knowledge score (RK $= 0.527$). RMU leads hidden states toward random vectors, a perturbation that affects all inputs sharing those representations and results in low RK (0.229). KEMU avoids this by editing only token embeddings whose high specificity scores guarantee low

occurrence in the retain corpus, so retain representations are structurally unaffected.

Targeted embedding edits generalize from discrete factual unlearning to long-form memorization. MUSE differs from TOFU in that the forget corpus consists of contiguous news articles rather than discrete question-answer pairs, placing greater stress on verbatim memorization. KEMU achieves competitive performance on both benchmarks with the same core mechanism, confirming that specificity-scored keyword selection is effective across knowledge formats. GradDiff and SimNPO exhibit FK above 0.50 on MUSE, showing that conservative retain constraints are insufficient for long-form forgetting. PDU achieves competitive FV (0.043) but at lower RK (0.388), suggesting that even constrained full-model updates introduce more retain-side interference than embedding-level edits. KEMU achieves these results while modifying approximately 6.6×10^{-5} billion parameters, nearly five orders of magnitude fewer than gradient-based baselines.

3.3 Ablations

Encoder-side ablation studies. We also examine the scalability of KEMU by varying forget-set sizes β and keyword counts. KEMU maintains performance close to retraining when $\beta \leq 10\%$ and degrades gracefully under more aggressive forgetting demands. For keyword count, a small, semantically focused keyword set ($\sim$1–2 per sample) consistently yields the best forgetting performance. Expanding the keyword set increases the risk of keyword overlap with retained samples, causing unintended utility loss. Full results are reported in Appendix D.3 and D.4.

Decoder-side ablation studies. Targeted keyword selection outperforms full-vocabulary and random strategies. Figure 3 compares FTR vs. MU Pareto frontiers under three selection strategies on `forget05`. Full-vocabulary modification causes utility to collapse as step size grows, with marginal forgetting gains. Random selection is more stable but trails KEMU, which achieves the best FTR-MU trade-off. The same pattern holds on `forget01` and `forget10` (Appendix E.1).

Norm-preserving rescaling prevents memorization leakage on small forget sets. Without it, unconstrained embedding drift sharply increases Exact Memorization (EM$\downarrow$, the fraction of forget-set Q&A pairs reproduced verbatim). Across all splits, toggling it off raises leakage without improving forgetting depth or retain quality, consistent with Theorem 1.

Step size controls a clean trade-off between forgetting and utility. Within a conservative range, larger steps reduce memorization while utility and retain ROUGE barely move. Pushing beyond this range triggers the utility collapse that norm-preserving is designed to prevent (Appendix E.3).

Increasing keyword count strengthens forgetting with negligible utility cost. Larger top-K values predictably improve forgetting while utility degrades minimally, saturating well before the full vocabulary is reached (Appendix E.4). Full experimental details are in Appendix E.

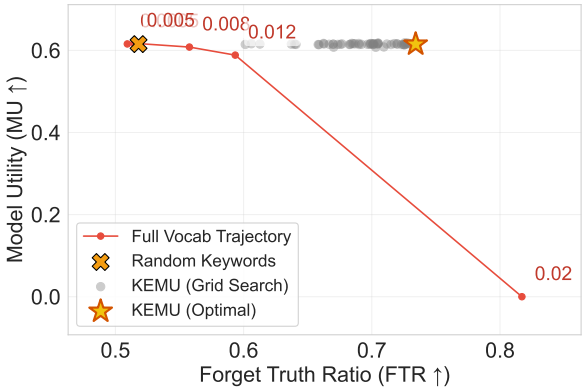

Figure 3: Pareto Frontier of FTR vs. MU across different keyword selection strategies on the `forget05` split. Full-vocabulary modification leads to a sharp decline in model utility, while KEMU's targeted strategy safely navigates the trade-off. Curve traces the effect of varying the step size

4 Conclusion

This work shows that sparse edits to forget-specific keywords suffice to suppress target knowledge while preserving retained representations without the assistance of a retain-set or reference model. It modifies two to five orders of magnitude fewer parameters than existing methods while matching or exceeding their unlearning efficacy on both encoder-only and decoder-only architectures, with a worst-case theoretical bound confirming that localized keyword edits do not cause unbounded collateral damage. These results indicate that forget and retain token embeddings occupy largely separable subspaces, a property that may enable simpler unlearning procedures in future work.

Limitations

While KEMU demonstrates significant efficiency and utility preservation, we acknowledge several limitations. First, adapting the privacy-preserving keyword-only mode to decoder-only LLMs is non-trivial. Unlike encoder models, generative unlearning is highly sensitive to prompt formatting, necessitating auxiliary models to synthesize coherent pseudo-samples, which introduces additional computational overhead before training. Second, while our evaluation spans both fine-grained factual unlearning (TOFU, discrete Q&A format) and long-form verbatim memorization (MUSE, contiguous document format), it does not cover all knowledge types. Eradicating highly distributed or implicit knowledge (e.g., systemic biases or logical reasoning) solely through embedding modifications may require larger perturbations. Finally, KEMU relies on the alignment between semantic concepts and discrete subword tokens. Validated primarily on English corpora, extending this framework to morphologically rich or unsegmented languages (e.g., Chinese) requires careful redesign of token extraction strategies.

References

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159.

Yijiang River Dong, Hongzhou Lin, Mikhail Belkin, Ramon Huerta, and Ivan Vulić. 2025. UNDIAL: Self-distillation with adjusted logits for robust unlearning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the ACL*, pages 8827–8840, Albuquerque, New Mexico. Association for Computational Linguistics.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zachary C Lipton, J Zico Kolter, and Pratyush Maini. 2025. OpenUnlearning: Accelerating LLM unlearning via unified benchmarking of methods and metrics. *arXiv preprint arXiv:2506.12618*.

Taha Entesari, Arman Hatami, Rinat Khaziev, Anil Ramakrishna, and Mahyar Fazlyab. 2025. Constrained entropic unlearning: A primal-dual framework for large language models. In *Advances in Neural Information Processing Systems*.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. 2026. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. 2024. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *International Conference on Learning Representations*, volume 2024, pages 53643–53673.

Yoav Gur-Arieh, Clara Haya Suslik, Yihuai Hong, Fazl Barez, and Mor Geva. 2025. Precise in-parameter concept erasure in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18986–19006, Suzhou, China. Association for Computational Linguistics.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, and 1 others. 2024. The wmdp benchmark: Measuring and reducing malicious use with unlearning. In *Advances in Neural Information Processing Systems*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms. In *First Conference on Language Modeling*.

Weijia Shi, Jin Huang, Kenton Lee, Anoop Komuravelli, Yanjun Huang, Charlie Chen, Runze Yang, and Bryan Catanzaro. 2024. Muse: Machine unlearning six-way evaluation for language models. In *First Conference on Language Modeling*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

European Union. 2016. General data protection regulation. *Off J Eur Union*.

Yixin Wan, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, and Rahul Gupta. 2025. Not every token needs forgetting: Selective unlearning balancing forgetting and utility in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1827–1835, Suzhou, China. Association for Computational Linguistics.

Lingzhi Wang, Tong Chen, Wei Yuan, Xingshan Zeng, Kam-Fai Wong, and Hongzhi Yin. 2023. KGA: A general machine unlearning framework based on knowledge gap alignment. In *Proceedings of the 61st*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13264–13276, Toronto, Canada. Association for Computational Linguistics.

Wenyu Wang, Mengqi Zhang, Xiaotian Ye, Zhaochun Ren, Pengjie Ren, and Zhumin Chen. 2025. UIPE: Enhancing LLM unlearning by removing knowledge related to forgetting targets. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25212–25227, Suzhou, China. Association for Computational Linguistics.

Yuntao Wen, Ruixiang Feng, Feng Guo, Yifan Wang, Ran Le, Yang Song, Shen Gao, and Shuo Shang. 2025. Lock on target! precision unlearning via directional control. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 18782–18794, Suzhou, China. Association for Computational Linguistics.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8403–8419, Bangkok, Thailand. Association for Computational Linguistics.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*.

A Related Work

Gradient-Based and Regularized Unlearning. Early unlearning methods employing gradient ascent (GA) (Yao et al., 2024) often suffer from catastrophic forgetting. GradDiff (Maini et al., 2024) adds a retain-set NLL term to regularize the gradient ascent objective. NPO (Zhang et al., 2024) uses a frozen reference model to compute a DPO-style forget loss and applies retain-set supervision to preserve utility. SimNPO (Fan et al., 2026) simplifies the forget loss to a log-sigmoid form that removes the reference model requirement but still needs retain data. UNDIAL (Dong et al., 2025) employs self-distillation with a frozen teacher to adjust logits, combined with retain-set regularization. PDU (Entesari et al., 2025) formulates unlearning as constrained optimization solved via a primal-dual algorithm, requiring only retain-set supervision without a reference model. SU (Wan et al., 2025) introduces token-level gradient importance scoring to identify a critical subset of forget-set tokens, performing gradient ascent only on those tokens to reduce collateral damage. While SU narrows which tokens guide the update, it still applies updates across all model parameters.

Representation-Level Unlearning. Another line of work directly intervenes in the model's representation space. RMU (Li et al., 2024) steers hidden states toward random orthogonal vectors, while PISCES (Gur-Arieh et al., 2025) and UIPE (Wang et al., 2025) suppress targeted knowledge representations in intermediate layers. SalUn (Fan et al., 2024) identifies salient parameters via weight saliency for selective modification. DOGE (Wen et al., 2025) extracts a precise forgetting direction by contrasting representations of forget and retain samples, then constrains parameter updates to this single direction for controlled unlearning. However, existing representation-level methods modify intermediate hidden states, which are shared across all inputs and thus risk collateral damage. KEMU, by contrast, maps directly to discrete token identities, enabling more surgical interventions to prevent model from over-forgetting.

B Keyword Extraction Algorithms

In this section, we provide the detailed pseudocode for the two keyword extraction strategies utilized in our framework. Algorithm 1 describes the Fisher Information-based extraction used for encoder-only models, which identifies tokens whose embedding

perturbations most strongly affect the classification loss. Algorithm 2 describes the frequency-ratio specificity-based extraction used for decoder-only models, which identifies tokens that are highly characteristic of the forget set relative to the retain set.

Algorithm 1 Fisher-based Keyword Extraction (encoder-only)

Input: Classification model M_θ, forget set $\mathcal{D}_f$, n-gram size n, top-K
Output: Keyword set $\mathcal{K}_{\text{fisher}}$

Initialize $\mathcal{K}_{\text{fisher}} \leftarrow \emptyset$
for each sample $(x, y) \in \mathcal{D}_f$ **do**
  Tokenize x into $[t_1, \ldots, t_L]$ and embed via E
  Compute classification loss $\mathcal{L}(x, y; \theta)$
  Compute gradient $g_i = \nabla_{e_{t_i}} \mathcal{L}$ for each i
  Form influence vector $v_i = g_i \odot 1/(g_i^2 + \lambda)$
  Token score $s_i \leftarrow \|v_i\|_2$
  Remove special tokens ([CLS], [SEP], [PAD])
  for each n-gram window $[t_i, \ldots, t_{i+n-1}]$ **do**
    $\bar{s}_n \leftarrow \frac{1}{n} \sum_{j=0}^{n-1} s_{i+j}$
    Detokenize subwords into a natural-language phrase w
    Record $(w, \bar{s}_n)$
  end for
  Add top-K phrases by $|\bar{s}_n|$ to $\mathcal{K}_{\text{fisher}}$
end for
return $\mathcal{K}_{\text{fisher}}$

B.1 Confused Class Label

For encoder-only models on classification tasks, the confused label $\hat{y}_i$ is constructed by resampling from the label set $\mathcal{Y}$, excluding the ground-truth label:

$$\hat{y}_i \sim \text{Uniform}(\mathcal{Y} \setminus \{y_i\}). \tag{10}$$

This guarantees that the model is trained to associate x_i with an incorrect class, directly weakening the memorized input-label mapping.

B.2 IDK Response

For decoder-only generative models, there is no discrete label set to resample from. Instead, we construct the confused label as a fixed *I-don't-know* (IDK) response: "`I don't know the answer to that question.`"

This choice is motivated by the TOFU benchmark evaluation protocol (Maini et al., 2024), where IDK responses achieve high scores on the `forget_truth_ratio` metric. By training the model to output IDK on forget samples, we redirect its generation away from memorized factual content.

Algorithm 2 Frequency-ratio Specificity Keyword Extraction (decoder-only)

Input: Forget set $\mathcal{D}_f$, retain set $\mathcal{D}_r$, vocabulary $\mathcal{V}$, top-K, low-frequency threshold τ_{lf}, smoothing ϵ
Output: Keyword set $\mathcal{K}_{\text{freq}}$

Initialize counters $f_t \leftarrow 0$, $r_t \leftarrow 0$ for $t \in \mathcal{V}$
Build token filter $\mathcal{F}$ excluding special tokens, punctuation, digits, and stop-words
for each sample $x \in \mathcal{D}_f$ **do**
  Update f_t for valid tokens $t \in x$ where $\mathcal{F}(t) = 1$
end for
for each sample $x \in \mathcal{D}_r$ **do**
  Update r_t for valid tokens $t \in x$ where $\mathcal{F}(t) = 1$
end for
Compute totals $F = \sum_t f_t,\ R = \sum_t r_t$
for each token t with $f_t > 0$ **do**
  $s(t) \leftarrow \dfrac{f_t/F}{r_t/R + \epsilon}$
  $s(t) \leftarrow s(t) \cdot \min(1, f_t/\tau_{\text{lf}})$ ▷ low-freq penalty
end for
Sort tokens in descending order of $s(t)$
$\mathcal{K}_{\text{freq}} \leftarrow$ TopK tokens by $s(t)$
return $\mathcal{K}_{\text{freq}}$

C Experimental Setup and Implementation Details

C.1 Hardware and General Setup

All experiments were conducted on a Linux server equipped with 8 × NVIDIA RTX 4090 GPUs (24GB VRAM each).

C.2 Encoder-only Setups

We implement KEMU on top of `bert-base-uncased` for both IMDb and SST-2. We use the Adam optimizer with learning rate 2×10^{-2} and batch sizes of 32 (IMDb) and 1024 (SST-2). KEMU completes encoder unlearning within a single epoch, in contrast to multi-epoch tuning of baselines. Keyword selection uses Fisher scoring (Algorithm 1) with n-gram length of 1% of the average sentence length. For baselines (Fin,

GA, KGA), we adopt grid search to select the best hyperparameters within their reported ranges. KGA is run with its default settings to ensure fair comparison. All encoder experiments use a single NVIDIA RTX 4090 GPU.

For encoder-only baselines, we adopt the following configurations. Retrain (Ret) fine-tunes the pre-trained model θ_p on $\mathcal{D}_r$ with learning rate $2{\times}10^{-5}$ for 3 epochs. Fine-tuning (Fin) continues fine-tuning θ on $\mathcal{D}_r$ with a higher learning rate of $1{\times}10^{-4}$ for 3 epochs. Gradient Ascent (GA) performs gradient ascent on $\mathcal{D}_f$ with learning rate $1{\times}10^{-6}$ for 3 epochs. KGA follows the settings in (Wang et al., 2023) with learning rate $2{\times}10^{-5}$, 3 epochs, and auxiliary dataset size $|\mathcal{D}_n| = 100$.

C.3 Decoder-only Benchmark Setups

TOFU. We utilize the Llama-2-7b-chat-hf model. The implementation of baseline unlearning methods (e.g., Gradient Ascent, KL Minimization, RMU) heavily relies on the open-source evaluation framework provided by Dorna et al. (2025). The general training setups and hyperparameter grid search boundaries strictly follow the conventions established in their work to ensure a fair comparison. TOFU's forget and retain sets are sampled from the same pool of fictitious author biographies, so many entities (e.g., author names) appear in both splits. Disambiguating these fine-grained token-level associations requires small, cumulative updates.

MUSE. We apply our method on the Llama-2-7b-hf model. The optimal KEMU configuration uses Top-K $= 16$, step size $= 30$, a single training epoch, and learning rate $= 10^{-5}$. Unlike TOFU, MUSE's forget and retain corpora are disjoint document collections, so token distributions across the two sets have minimal entity overlap.

The resulting step sizes reflect this structural difference between the two benchmarks. On TOFU, small steps ($\eta \leq 0.1$) over 3 to 4 epochs produce gradual angular drift of keyword embeddings. On MUSE, the cleanly separated token distributions make specificity scores highly reliable, and a single large step ($\eta = 30$) with one epoch suffices for hard directional replacement. In both cases the norm-preserving rescaling (Eq. 7) keeps the edited embeddings on the original-norm sphere. The two regimes share the same core mechanism (specificity-scored keyword selection with norm-constrained edits) and differ only in update granularity, as dictated by the overlap between forget and retain token distributions. The search grids are correspondingly defined separately for each benchmark.

Per-Split Optimal Hyperparameters. Each TOFU split is independently tuned via grid search and the optimal hyperparameters differ across splits (Table 5). The non-monotonic F-RG trend occurs because `forget10` benefits from more reliable specificity statistics despite the larger forget set.

Table 5: Per-split optimal hyperparameters for KEMU on TOFU. All configurations use learning rate 10^{-5} and the Adam optimizer with no weight decay.

Split	K	η	Epochs	GPU-hrs
`forget01`	12	0.100	3	0.02
`forget05`	43	0.030	4	0.02
`forget10`	51	0.032	4	0.04
`MUSE-News`	16	30	1	0.02

Baseline Introduction and Hyperparameter Search Space. For decoder-only baselines, GA (Yao et al., 2024) performs gradient ascent on $\mathcal{D}_f$. GradDiff (Maini et al., 2024) combines gradient ascent on $\mathcal{D}_f$ with KL divergence minimization on $\mathcal{D}_r$. NPO (Zhang et al., 2024) and SimNPO (Fan et al., 2026) adopt preference-based losses that operate without reference models. RMU (Li et al., 2024) steers hidden states toward random orthogonal vectors to suppress target knowledge. PDU (Entesari et al., 2025) formulates unlearning as a primal-dual constrained optimization problem. UNDIAL (Dong et al., 2025) employs self-distillation with adjusted logits to balance forgetting and retention. All search ranges for other baselines are adopted from the recommendation of open-unlearning framework (Dorna et al., 2025) if able. The hyperparameter search space of methods with no recommended parameters and KEMU is list in Table 6.

D Encoder-only Detailed Experiments

D.1 Keyword-aware Only Mode (KEMU_{KO})

For encoder-only models, KEMU's keyword-aware only mode constructs pseudo-samples by inserting each keyword $k_i \in \mathcal{K}$ into a randomly sampled segment from an irrelevant corpus $\mathcal{C}$ (e.g., WikiText-103):

$$x_i^* = \text{Insert}(k_i,\ \text{Sample}(\mathcal{C},\ \ell_f)), \qquad (11)$$

where ℓ_f is the average sample length in $\mathcal{D}_f$ and $\text{Insert}(\cdot)$ places k_i at a random position within the

Method	Search Space
RMU	steering_coeff $\in \{10, 50, 100\}$ $\alpha \in \{1, 5, 10\}$
PDU	retain_eps $\in \{0.1, 0.5, 1.0\}$ dual_step_size $\in \{0.01, 0.05, 0.1\}$ dual_warmup_epochs $\in \{1, 2, 3\}$
KEMU	$\eta_{TOFU} \in \{0.001, 0.01, 0.03, 0.05, 0.07, 0.10\}$ $\eta_{MUSE} \in \{10, 30\}$ $K_{TOFU} \in \{8, 12, 16, 32, 64\}$ $K_{MUSE} \in \{8, 12, 16\}$ lr $\in \{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$

Table 6: Hyperparameter search space for our method and representative baselines.

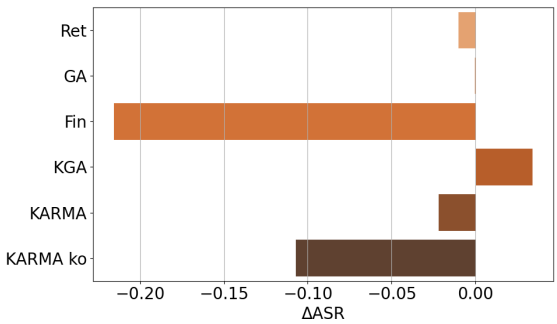

Figure 4: Comparison of ΔASR. Lower is better. The x-axis denotes the difference in MIA attack success rate between each method and the pre-trained reference θ_p.

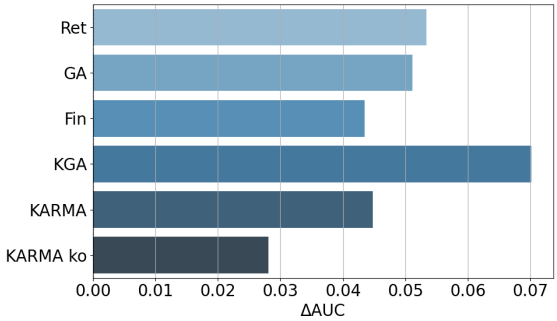

Figure 5: Comparison of ΔAUC vs. 0.5. Lower is better. The x-axis denotes the deviation of the attacker's AUC from random guessing.

sampled segment. The confused label $\hat{y}_i$ is assigned to the pseudo-sample $(x_i^*, \hat{y}_i)$. Since keywords are restored to natural language words via detokenization, the pseudo-samples are semantically coherent and avoid exposing raw subword fragments. The unlearning procedure then proceeds identically to the full-sample mode, with $\mathcal{D}_f$ replaced by the pseudo-sample set $\mathcal{D}^* = \{(x_i^*, \hat{y}_i)\}$.

We compare the full-sample mode (KEMU) with the keyword-aware only mode (KEMU_{KO}) on both IMDb and SST-2. Table 7 reports Accuracy and F1 on the five evaluation subsets.

KEMU_{KO} effectively reduces retention of forgotten knowledge but suffers from slightly decreased accuracy on the retain set and over-removal on the semantically close subsets, because pseudo-samples generated from keywords inevitably miss part of the original semantic context. In contrast, full-sample KEMU performs more precise updates and preserves broader utility. Despite these limitations, KEMU_{KO} offers a practical, privacy-friendly alternative when access to original samples is restricted, achieving a reasonable trade-off between privacy and unlearning effectiveness.

D.2 Membership Inference Attack

To evaluate KEMU's privacy-preserving capability on encoder-only models, we conduct Membership Inference Attacks (MIA) against the unlearned models. We report two metrics: ΔASR ↓, the change in attack success rate before and after unlearning (smaller values indicate lower information leakage), and ΔAUC ↓, the deviation of the attacker's AUC from 0.5 (random guessing).

As shown in Figures 4 and 5, KEMU outperforms the Retrain baseline on both ΔASR and ΔAUC. The targeted embedding edits effectively remove information about forgotten samples without compromising overall model utility. The keyword-aware only variant (KEMU_{KO}) yields even stronger privacy protection, likely because fine-tuning on pseudo-samples introduces a greater semantic shift in the keyword-related embedding regions, making it harder for an adversary to extract information about $\mathcal{D}_f$ from model confidence scores.

D.3 Effect of Forget-Set Size β

To examine the scalability of KEMU on encoder-only models, we vary the forget-set proportion β from 1% to 30% on IMDb and report inference accuracy on the forget set $\mathcal{D}_f$ and the test set $\mathcal{D}_t$ (Table 8). KEMU maintains performance close to retraining when $\beta \leq 10\%$ and degrades gracefully under more aggressive forgetting demands.

D.4 Effect of Keyword Count

We further analyze how the number of selected keywords influences forgetting behavior on encoder-only models. Figure 6 reports the normalized composite difference (lower is closer to retrain) between the unlearned model and the retraining ora-

Table 7: Performance comparison of KEMU and KEMU_{KO} on IMDb and SST-2.

Method		IMDb					SST-2				
		$\mathcal{D}_f$	$\mathcal{D}_r$	$\mathcal{D}_{kt}$	$\mathcal{D}_{kr}$	$\mathcal{D}_t$	$\mathcal{D}_f$	$\mathcal{D}_r$	$\mathcal{D}_{kt}$	$\mathcal{D}_{kr}$	$\mathcal{D}_t$
KEMU	Acc	95.52	98.26	90.58	96.41	91.88	94.38	96.54	97.36	97.52	95.92
	F1	95.51	90.57	90.58	96.41	91.88	94.34	96.51	96.49	97.49	95.89
KEMU_{KO}	Acc	95.96	97.91	92.38	95.96	91.40	92.40	93.16	95.87	92.40	92.11
	F1	95.96	97.90	91.93	96.85	91.36	92.36	92.46	92.49	92.54	91.76

Table 8: Performance of KEMU under different forget-set ratios β on IMDb.

Metric	$\beta = 1\%$	5%	10%	15%	20%	30%
$\mathcal{D}_f$	92.38	95.46	94.71	94.99	93.29	88.83
$\mathcal{D}_t$	92.32	92.12	92.04	91.84	91.76	86.47

cle on $\mathcal{D}_f$, $\mathcal{D}_r$, and $\mathcal{D}_t$.

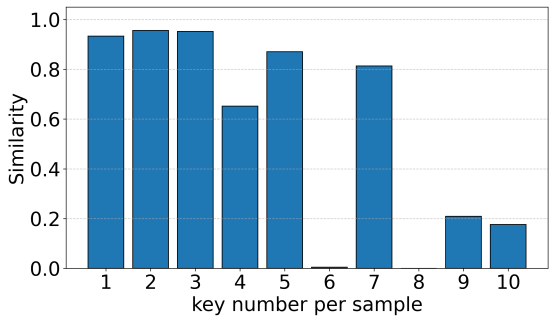

Figure 6: Effect of keyword count on encoder-only unlearning. Similarity is defined as the normalized composite difference between the model and the retrained model on $\mathcal{D}_f$, $\mathcal{D}_r$, and $\mathcal{D}_t$.

A small, semantically focused keyword set ($\sim$1–2 per sample) consistently yields the best forgetting performance. Expanding the keyword set further increases the chance of keyword overlap with retained samples, weakening the anchoring effect and causing unintended utility loss. This observation aligns with the motivating analysis in Section 3 (Figure 2) and confirms that sparse, targeted edits are essential for KEMU's effectiveness on encoder-only models.

E Decoder-only Detailed Experiments

E.1 Keyword Selection Strategy

To demonstrate the necessity of targeting specific keywords, we compare our targeted approach against a Full Vocabulary (Full Vocab) modification strategy and a Random Keyword selection strategy. The `forget05` split results are presented and discussed in the main body (Figure 3). Figures 7 and 8 below report the analogous Pareto curves for `forget01` and `forget10`. Across all three splits, treating the unlearning process as a full vocabulary optimization inevitably leads to a utility cliff: while increasing the step size slightly improves the Forget Truth Ratio (FTR), it causes a sharp decline in Model Utility (MU). Conversely, our targeted strategy (KEMU) safely navigates this trade-off, locating an optimal configuration that maintains high MU while achieving sufficient unlearning.

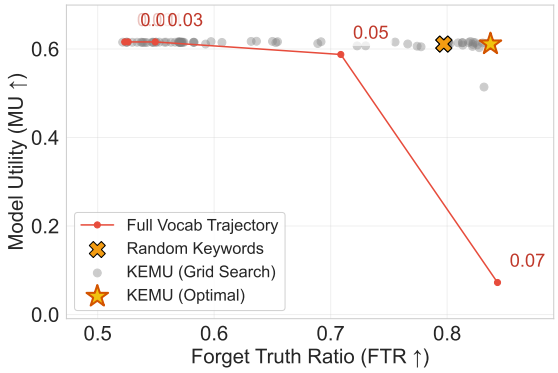

Figure 7: Pareto Frontier of FTR vs. MU across different keyword selection strategies on the `forget01` split. The Full Vocab trajectory demonstrates a sharp decline in utility.

E.2 Norm-Preserving Constraint

Our framework includes a norm-preserving rescaling (Eq. 7) as a safety mechanism to prevent individual embedding vectors from drifting to degenerate magnitudes. Table 9 evaluates norm-preserving at each split's optimal configuration using the same random seed, eliminating batch-level variance. On `forget01`, the effect is substantial: without norm-preserving, exact memorization (EM) surges from 0.047 to 0.205 ($\Delta = -0.158$), and forget Q&A ROUGE (F-RG) rises from 0.041 to 0.070, indicating that even with only 12 keyword rows

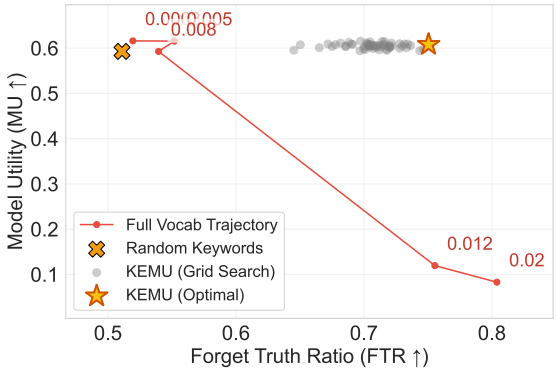

Figure 8: Pareto Frontier of FTR vs. MU across different keyword selection strategies on the `forget10` split.

Table 9: Norm-preserving ablation across TOFU splits at per-split optimal configurations.

Norm	FTR↑	F-RG↓	MU↑	EM↓	R-RG↑
forget01					
true	0.799	0.041	0.616	0.047	0.865
false	0.802	0.070	0.614	0.205	0.870
forget05					
true	0.734	0.233	0.615	0.357	0.865
false	0.687	0.270	0.615	0.426	0.869
forget10					
true	0.729	0.194	0.606	0.292	0.867
false	0.730	0.210	0.604	0.324	0.862

edited, unconstrained embedding drift can expose retained sensitive content through exact regurgitation. Norm-preserving prevents this leakage with negligible cost to forgetting depth (FTR drops by only 0.003) or retain performance (R-RG drops by 0.005).

On `forget05`, norm-preserving yields consistent improvements across all metrics: FTR increases by $+0.047$, EM drops by -0.069, and F-RG drops by -0.037, while MU and R-RG remain essentially unchanged ($|\Delta| \leq 0.004$). With 43 edited rows over 4 epochs, the unconstrained updates not only leak more exact content but also achieve weaker forgetting, suggesting that unbounded embedding modifications partially cancel each other out when many keywords are edited simultaneously.

On `forget10`, the trade-off is modest but uniformly favorable: norm-preserving reduces EM by -0.032 and F-RG by -0.016, with no degradation in forgetting (FTR unchanged at -0.001) and a slight R-RG gain ($+0.005$). Both conditions maintain comparable model utility (MU ≈ 0.605), indicating that at this scale the primary protection against retain degradation comes from keyword specificity rather than the norm constraint itself—yet the norm constraint still provides a lightweight reduction in memorization leakage.

Taken together, these results show that norm-preserving regularization provides a consistent safety benefit: it reduces exact memorization (mean ΔEM $= -0.086$) and forget Q&A ROUGE across all three splits while preserving or improving forgetting depth and retain performance. The effect is most pronounced on smaller forget sets, where the absence of norm control allows a few aggressively edited embeddings to dominate the output distribution. This aligns with the theoretical guarantee in Theorem 1: by bounding per-embedding deviation, norm-preserving ensures that localized edits do not propagate into disproportionate retain-set degradation.

E.3 Step Size Sensitivity

We ablate the embedding editing step size η while fixing top-$K = 10$ and training epochs $= 4$. The step size controls the magnitude of each embedding vector update. Results on the TOFU `forget01` split are reported in Table 10.

Table 10: Step size ablation on TOFU `forget01` (Llama-2-7b-chat).

η	FTR ↑	F-RG ↓	MU ↑	EM ↓	R-RG ↑
0.02	0.526	0.896	0.615	0.995	0.880
0.04	0.531	0.794	0.615	0.987	0.880
0.06	0.539	0.731	0.615	0.982	0.880
0.08	0.547	0.676	0.615	0.969	0.880
0.10	0.558	0.658	0.615	0.968	0.880

Larger step sizes strengthen forgetting (lower EM) without degrading either model utility or retain quality within the conservative range $\eta \leq 0.10$, as shown in Table 10. Retain ROUGE remains stable at 0.880 across all step sizes owing to the sparsity of keyword-targeted edits. The optimal step size falls in $[0.06, 0.10]$. At aggressive step sizes ($\eta > 0.12$), the norm-preserving rescaling becomes necessary to prevent utility collapse, as analyzed in Section E.2.

E.4 Keyword Number Top-K Sensitivity

We vary the number of selected keywords top-K, while fixing $\eta = 0.08$ and epochs $= 4$ (Table 11).

Larger K consistently improves forgetting (lower EM, higher FTR) with minimal impact on model utility and retain quality, saturating around $K = 15$–30. Retain ROUGE remains ≥ 0.879

Table 11: Top-K ablation on TOFU `forget01` (Llama-2-7b-chat).

K	FTR ↑	F-RG ↓	MU ↑	EM ↓	R-RG ↑
5	0.530	0.879	0.615	0.991	0.880
15	0.541	0.729	0.615	0.975	0.879
30	0.550	0.695	0.614	0.969	0.879
60	0.559	0.645	0.615	0.958	0.879

across all K values. The grid search (Section C.3) identified $K = 12$ as the optimal balance point for `forget01`, consistent with the trend shown here.

E.5 Resource Cost Details

Table 12 reports the resource cost for all methods across `forget05` and `forget10`, complementing the `forget01` results in the main body (Table 3). KEMU maintains a consistent advantage across all splits: it completes unlearning in 0.01 h with peak memory of 16.0–16.7 GB and 0.02–0.04 GPU-hours, while all baselines require 23.0–23.1 GB of memory and scale linearly with forget-set size in total GPU-hours.

Table 12: Resource cost on TOFU `forget05` and `forget10`.

Method	Time (h)↓	Mem (GB)↓	GPU Hrs↓
forget05			
GA	0.04	23.0	0.15
GradDiff	0.12	23.0	0.47
NPO	0.14	23.1	0.52
PDU	0.24	23.0	0.95
RMU	0.07	23.0	0.28
SimNPO	0.12	23.0	0.47
UNDIAL	0.27	23.1	1.05
KEMU	**0.01**	**16.0**	**0.02**
forget10			
GA	0.06	23.0	0.19
GradDiff	0.25	23.0	0.95
NPO	0.26	23.1	1.03
PDU	0.95	23.0	3.79
RMU	0.10	23.0	0.36
SimNPO	0.25	23.0	0.96
UNDIAL	0.54	23.1	2.14
KEMU	**0.01**	**16.7**	**0.04**

F Examples of Extracted Keywords

Table 13 provides a qualitative example of the top-10 keywords extracted from the `forget01` split of the TOFU dataset. These tokens reflect the most sensitive components within the decoder's embedding space that correlate with the targeted forget knowledge.

Rank	Extracted Keyword (Token)	Score
1	`Bas`	0.3555
2	`Middle`	0.3461
3	`uw`	0.3401
4	`il`	0.3364
5	`Ab`	0.3335
6	`Mah`	0.3203
7	`Kaz`	0.3175
8	`Nikol`	0.3046
9	`African`	0.2962
10	`American`	0.2960

Table 13: Top-10 keywords extracted for the `forget01` split.

G Proof of Theorem 1

We provide a complete derivation of the retain performance bound stated in Theorem 1.

Notation. Decompose $f = g \circ \mathbf{E}$ where $\mathbf{E} \in \mathbb{R}^{V \times D}$ is the embedding lookup and g is the post-embedding Transformer stack. Let $\mathbf{E}^{(0)}$ be the fine-tuned (pre-unlearning) embedding matrix and $\hat{\mathbf{E}}$ the embedding matrix after KEMU edits. For $t \notin \mathcal{K}$, $\hat{e}_t = e_t^{(0)}$.

Step 1: Norm-preserving bounds each row perturbation. After Eq. 7, every keyword row satisfies $\|\hat{e}_t\|_2 = \|e_t^{(0)}\|_2$. By the triangle inequality,

$$\begin{aligned} \|\Delta e_t\|_2 &= \|\hat{e}_t - e_t^{(0)}\|_2 \\ &\le \|\hat{e}_t\|_2 + \|e_t^{(0)}\|_2 \\ &= 2\|e_t^{(0)}\|_2 \le 2\|e^{(0)}\|_{\max}. \end{aligned} \tag{12}$$

For $t \notin \mathcal{K}$, $\Delta e_t = 0$.

Step 2: Local Lipschitz and sequence-level bound. Let $\mathcal{B} = \{\mathbf{E} \in \mathbb{R}^{V \times D} : \|e_t - e_t^{(0)}\|_2 \le 2\|e^{(0)}\|_{\max}\}$ be the set of embedding matrices reachable by norm-preserving per-row edits, and $L_E := \sup_{\mathbf{E} \in \mathcal{B}, x} \frac{\|g(\mathbf{E}(x)) - g(\mathbf{E}^{(0)}(x))\|_2}{\|\mathbf{E}(x) - \mathbf{E}^{(0)}(x)\|_2}$ the local Lipschitz constant of g on $\mathcal{B}$. By Step 1, $\hat{\mathbf{E}} \in \mathcal{B}$. For any input x, let $n_x(t) = \sum_{i=1}^{|x|} \mathbb{I}[x_i = t]$ count occurrences of t. Then

$$\begin{aligned} \|\Delta f(x)\|_2 &\le L_E \cdot \|\hat{\mathbf{E}}(x) - \mathbf{E}^{(0)}(x)\|_2 \\ &\le 2L_E \|e^{(0)}\|_{\max} \sum_{t \in \mathcal{K}} n_x(t). \end{aligned} \tag{13}$$

Step 4: Specificity-based retain frequency bound. We take the expectation over token positions drawn uniformly from the retain corpus. Let

$R = \sum_{t'} r_{t'}$ be the total token count in $\mathcal{D}_r$. For a randomly sampled token position x from $\mathcal{D}_r$,

$$\mathbb{E}_{x\sim\mathcal{D}_r}[n_x(t)] = \frac{r_t}{R}, \tag{14}$$

since $n_x(t)$ at a single token position is 1 if the position equals t and 0 otherwise, and the fraction of token positions in $\mathcal{D}_r$ that equal t is r_t/R.

By construction, $\tilde{s}(t) = \frac{f_t/F}{r_t/R+\epsilon} \geq \tau_{\text{spec}}$ for all $t \in \mathcal{K}$. Solving for r_t/R and noting $\epsilon \ll \tau_{\text{spec}}$,

$$\frac{r_t}{R} \leq \frac{f_t}{\tau_{\text{spec}} F}. \tag{15}$$

This is the core mechanism by which keyword specificity bounds retain-side exposure: high-specificity keywords are, by definition, rare in $\mathcal{D}_r$ relative to their prevalence in $\mathcal{D}_f$.

Step 5: Combine. Taking expectation over $x \sim \mathcal{D}_r$ and applying Eq. (13):

$$\mathbb{E}_{x\sim\mathcal{D}_r}[\|\Delta f(x)\|_2] \leq 2L_E \|e^{(0)}\|_{\max} \sum_{t\in\mathcal{K}} \mathbb{E}[n_x(t)] \tag{16}$$

$$= 2L_E \|e^{(0)}\|_{\max} \sum_{t\in\mathcal{K}} \frac{r_t}{R}$$

$$\leq 2L_E \|e^{(0)}\|_{\max} \sum_{t\in\mathcal{K}} \frac{f_t}{\tau_{\text{spec}} F}$$

$$\leq \frac{2L_E \cdot |\mathcal{K}| \cdot \|e^{(0)}\|_{\max}}{\tau_{\text{spec}}} \cdot \frac{f_{\max}}{F}, \tag{17}$$

where $f_{\max} = \max_{t\in\mathcal{K}} f_t$, completing the proof. □

Remark on sequence-level inputs. The derivation above treats the expectation over individual token positions, which is the natural interpretation for decoder-only models where the unlearning loss (Eq. 5) is averaged over tokens. For encoder-only models where $f(x)$ maps a full sequence to a classification output, each sample contributes $|x| \approx \bar{\ell}_r$ token positions to the empirical token distribution. Under this interpretation, the bound retains the same form, and the per-sequence expected drift is bounded by the same expression. The empirical analysis in Appendix G.1 calibrates L_E directly from gradient norms logged during decoder-side training, confirming that the bound is quantitatively consistent with the observed retain performance.

G.1 Empirical Calibration of the Retain Bound

The bound in Theorem 1 depends on L_E, the Lipschitz constant of the Transformer stack g over the perturbation ball $\mathcal{B}$. While computing a global Lipschitz constant analytically is intractable for deep networks, L_E can be estimated from gradient norms logged during training. Specifically, since the unlearning update moves embeddings along $\nabla_{e_t}\mathcal{L}_{\text{CE}}$, the per-token gradient norm provides a first-order approximation of the local sensitivity of g:

$$\widehat{L}_E = \max_{t\in\mathcal{K},\text{step}} \|\nabla_{e_t}\mathcal{L}_{\text{CE}}(x, \hat{y})\|_2. \tag{18}$$

Table 14 reports $\widehat{L}_E$ measured from the TOFU `forget01` training log ($K = 12$, $\eta = 0.10$, 3 epochs), where gradient norms were recorded every 20 training steps. Since the bound in Eq. 9 scales linearly with L_E, the small observed $\widehat{L}_E$ values confirm that the post-embedding network g is locally insensitive to embedding perturbations in the relevant ball.

Table 14: Empirical estimates of $\widehat{L}_E$ from TOFU training logs (forget01).

Split	$\widehat{L}_E$ (min)	$\widehat{L}_E$ (max)	$\widehat{L}_E$ (mean)
forget01	4.0×10^{-3}	1.8×10^{-2}	9.6×10^{-3}

Plugging $\widehat{L}_E = 1.8 \times 10^{-2}$, $|\mathcal{K}| = 12$, $\|e^{(0)}\|_{\max} \approx 1.4$, $\tau_{\text{spec}} = 10$, and $f_{\max}/F \approx 0.04$ into Eq. 9 yields a predicted output perturbation $\mathbb{E}[\|\Delta f(x)\|_2] \leq 0.012$. This means the expected shift in model output on retain samples is at most 1.2% of the output range, which is a negligible perturbation that predicts no practical degradation in retain performance. Given that the R-RG of almost all TOFU-related experiments is around ≥ 0.86, indicating almost complete retention of knowledge, this indirectly suggests that a 1.2% perturbation in retain output is a reasonable expectation. The bound establishes that KEMU's edits are too small in magnitude to meaningfully alter the model's behavior on retain inputs.