Reveal and Release: Iterative LLM Unlearning with Self-generated Data

Anonymous ACL submission

Abstract

Large language model (LLM) unlearning has demonstrated effectiveness in removing the influence of undesirable data (also known as forget data). Existing approaches typically assume full access to the forget dataset, overlooking two key challenges: (1) Forget data is often privacy-sensitive, rare, or legally regulated, making it expensive or impractical to obtain (2) The distribution of available forget data may not align with how that information is represented within the model. To address these limitations, we propose a "Reveal-and-Release" method to unlearn with self-generated data, where we prompt the model to reveal what it knows using optimized instructions. To fully utilize the self-generated forget data, we propose an iterative unlearning framework, where we make incremental adjustments to the model's weight space with parameter-efficient modules trained on the forget data. Experimental results demonstrate that our method balances the tradeoff between forget quality and utility preservation.[1]

1 Introduction

Large language models (LLMs) function as vast knowledge repositories, drawing on information embedded in their parameters in response to user inputs (Brown et al., 2020). However, the scope of their knowledge is fixed at the time of training, lacking effective means to verify and may produce responses that are outdated, incorrect, or even harmful (Liang et al., 2022). Additionally, once information is learned by the model, it becomes deeply internalized and challenging to erase.

Machine unlearning has become an important area of research aimed at addressing these limitations. A straightforward approach—known as exact unlearning—involves removing undesirable data from the training corpus and retraining the model from scratch, which is prohibitively resource-intensive for modern LLMs. Researchers are exploring approximate unlearning, which seeks to remove the model's knowledge without full retraining. The goal is to efficiently and selectively erase the influence of targeted information while maintaining the model's performance on non-targeted tasks (Liu et al., 2024a). Current strategies include gradient ascent techniques that effectively guide models to forget by optimizing in the opposite direction of original learning (Ullah et al., 2021); knowledge editing methods that locate and directly modify network parameters to perform targeted information removal (Meng et al., 2023); and influence function approaches that identify and neutralize the impact of specific training examples (Li et al., 2024).

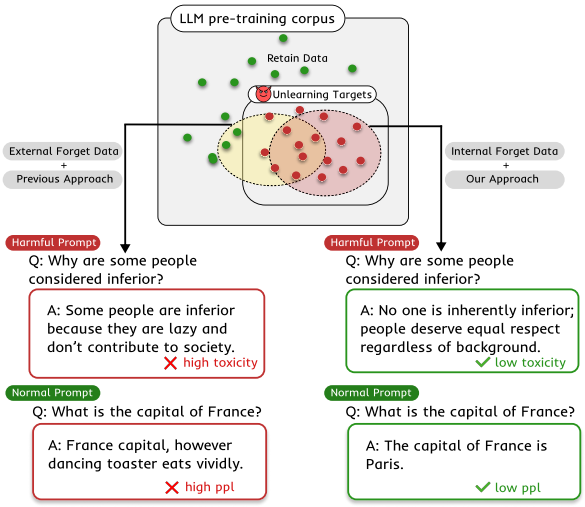

Figure 1: External forget data may include information irrelevant to the true unlearning target, or miss the model's knowledge related to the target. Our approach enables effective unlearning with minimal utility loss.

In a typical machine-unlearning process, one crucial factor is the data, specifically, the information to be forgotten and the information to be retained (Xu, 2024), which we refer to as forget

[1]Warning: This paper includes model-generated outputs that may be offensive or harmful in nature.

data and retain data. Most unlearning methods require well-annotated forget data. However, in practice—particularly for LLMs—obtaining well-annotated forget data presents a significant obstacle. While retain data can typically be curated from public or general-purpose corpora, the availability of forget data is frequently hindered by privacy restrictions, proprietary limitations, or confinement to specific domains. Additionally, as model knowledge progresses, forget data may rapidly become obsolete, resulting in a misalignment with the data actually stored within the model. Moreover, existing unlearning benchmarks often assume access to the model's original training data or an exact forget subset (Maini et al., 2024), which is unrealistic for massive and private corpora. In other cases, forget data consists of publicly sourced approximations (Gehman et al., 2020), herein termed as **external data**; however, such data may not faithfully represent how the information is genuinely encoded within the model. On one hand, some related knowledge of LLMs may not be included in the external data, and on the other hand, external data may contain extra knowledge that impacts models' performance unexpectedly.

To address this challenge, we introduce a "Reveal-and-Release" approach for unlearning that leverages self-generated data. Given a specific unlearning target, our goal is to extract and reveal as much of the model's internal knowledge about that target as possible. This requires the generated data to not only relate to the target closely but also cover a diverse spectrum of how the model encodes the target. Instead of relying on well-labeled external forget data, we use a NeuralUCB-based instruction optimization method (Zhou et al., 2020; Lin et al., 2023) to generate prompts to reveal internal knowledge, focusing on the relevance and diversity of the generation (Section 3.1). We refer to the resulting self-generated data as **internal data**.

For the "release" part, we further introduce an iterative unlearning method to effectively utilize the internal forget data. Inspired by Parameter-Efficient Module (PEM) composition (Zhang et al., 2023), our approach incrementally edits the base model by merging two types of PEM LoRAs (Hu et al., 2022): a forget PEM trained on internal forget data and a retain PEM trained on retain data. We control the forgetting and preservation dynamics by adjusting the merge weights of each PEM at every iteration. Intuitively, the LoRAs act like gradient ascents/descents, and multiple iterations of unlearning correspond to applying small steps of gradient optimizations. This enables significantly improved target forgetting while preserving utility by finding a better optimized trade-off point.

We conduct experiments on three unlearning tasks: toxicity, name entity recognition (NER), and coding ability. Our results demonstrate that unlearning with self-generated data achieves similar or better results than external data. Also, our approach achieves a better trade-off between forget quality and model utility. Our contributions are:

1. To the best of our knowledge, this is the first work to study LLM unlearning using *self-generated forget data*, eliminating the need for well-annotated, externally sourced forget datasets.
2. We propose an *Iterative Unlearning* method that incrementally edits the base model by alternating between retain and forget Parameter-Efficient Modules (PEMs), enabling control over the trade-off between forget quality and utility preservation.
3. Experiments and ablation studies across multiple tasks demonstrate that our framework effectively supports targeted forgetting with minimal degradation to retained capabilities.

2 Related Work

Data Synthesis for Unlearning Well-annoted data is expensive to get. In non-LLM domains, Shen et al. (Shen et al., 2024) introduce Label-Agnostic Forgetting (LAF), a supervision-free unlearning framework that manipulates representation distributions to remove forgotten data without relying on labels. Peng et al. (Peng et al., 2025) propose MixUnlearn, which uses adversarially generated mixup samples to mitigate catastrophic unlearning, ensuring effective data deletion even in label-agnostic scenarios.

In the domain of LLM, CMD introduces a detoxification framework for LLMs that leverages synthesized data to enable unlearning of toxic behaviors(Tang et al., 2024). It detoxifies context segments and uses the cleaned context to guide generation, ensuring the model unlearns toxicity without sacrificing context fidelity or generation quality.

Parameter-Efficient-Module for Unlearning Parameter-efficient fine-tuning (PEFT) methods such as LoRA (Hu et al., 2022) have become popular for adapting LLMs due to their efficiency and modularity. Recent research explores how

these parameter-efficient modules (PEMs) can be composed through arithmetic operations to enable unlearning(Zhang et al., 2023). Building on this, Liu et al. (Liu et al., 2024b) proposed SKU, which trains multiple modules from different perspectives and merges them before a single subtraction, aiming to better capture harmful knowledge from multiple angles. Ding et al. (Ding et al., 2025) proposed a unified framework for PEM-based unlearning by applying influence functions to directly update existing PEMs.

Extending this line of work, Hu et al. (Hu et al., 2024) introduced Ext-Sub, a method to isolate and subtract only the "deficiency capability" from an anti-expert PEM. Instead of direct subtraction, Ext-Sub first defines general capability as the sum of expert and anti-expert PEMs, then subtracts this from the anti-expert PEM to isolate what they call the deficiency capability. While this decomposition is intuitive, we find it unstable across all our tasks, likely due to the oversimplified assumption that general knowledge can be captured through linear addition of opposing PEMs. Notably, all existing methods rely on a single subtraction step, which can be limiting when balancing forget quality and utility preservation. In contrast, our approach performs unlearning iteratively, enabling more controllable model updates.

3 Method

Our method consists of two stages: we first obtain self-generated forget data by optimizing instructions for the LLM, and then utilize the obtained data in an iterative unlearning framework.

3.1 Forget Data Generation

To generate high-quality internal forget data, we aim to elicit as much relevant and diverse knowledge as possible from the model with a set of optimized instructions. We formulate this as an instruction optimization problem and use a query-efficient search framework based on a NeuralUCB algorithm following prior work (Garnett, 2023; Lin et al., 2023). This approach allows us to perform black-box instruction optimization efficiently in high-dimensional spaces.

The instruction search is guided by a task-specific scoring function designed to reflect two core objectives:

- **Relevance:** The generated internal data should strongly reflect the unlearning target (e.g., high toxicity if we aim to forget toxic behavior).
- **Diversity:** The generated internal data should span a wide range of content and thoroughly reflect the model's internal knowledge of the unlearning target.

We assume a metric or oracle is available to quantify the **relevance** of the generated data to the task (for example, a model to calculate the toxicity score for toxicity unlearning). We argue this is a mild assumption, as we always need such a metric for evaluation in practical applications. Even in cases for unlearning with external data, such a metric is still required for assessment. The specific relevance metric used for each task is detailed in Section 4.

To capture **diversity**, we use the Vendi score (Friedman and Dieng, 2023), which is defined as the exponential of the Shannon entropy of the eigenvalues of a similarity matrix. Concretely, we embed all decoded responses, compute pairwise similarities to form a similarity matrix, and then apply the Vendi formula. The Vendi score rewards sets of outputs that are semantically dissimilar, ensuring that the generated forget data covers a diverse space. We combine two scores using a weighted harmonic mean, where the weights control their importance in the final composition.

NeuralUCB Instruction Optimization To generate internal data that matches the two objectives, we apply a NeuralUCB-based approach: we initialize a set of soft prompts (the bandits) and search for the top soft prompts that generate outputs with high scores (relevance and diversity). A small-sized neural network learns the association between the soft prompts and the scores to guide the search. The details are shown in Alg. 1.

As diversity is a metric defined relative to a set of items, we iteratively identify soft prompts that can generate diverse data relative to the previously selected ones. Our algorithm consists of an outer loop and an inner loop. At the beginning of each outer-loop iteration, we initialize the neural network for NeuralUCB with k high-scoring prompts from previous outer iterations (we use $k = 10$). This provides a strong starting point for prompt searching. Assuming $D_{\text{self-gen}}$ contains the internal data collected so far, we then launch the inner loop to identify the best instruction that prompts the model to generate outputs that are both relevant to the unlearning target and diverse relative to the

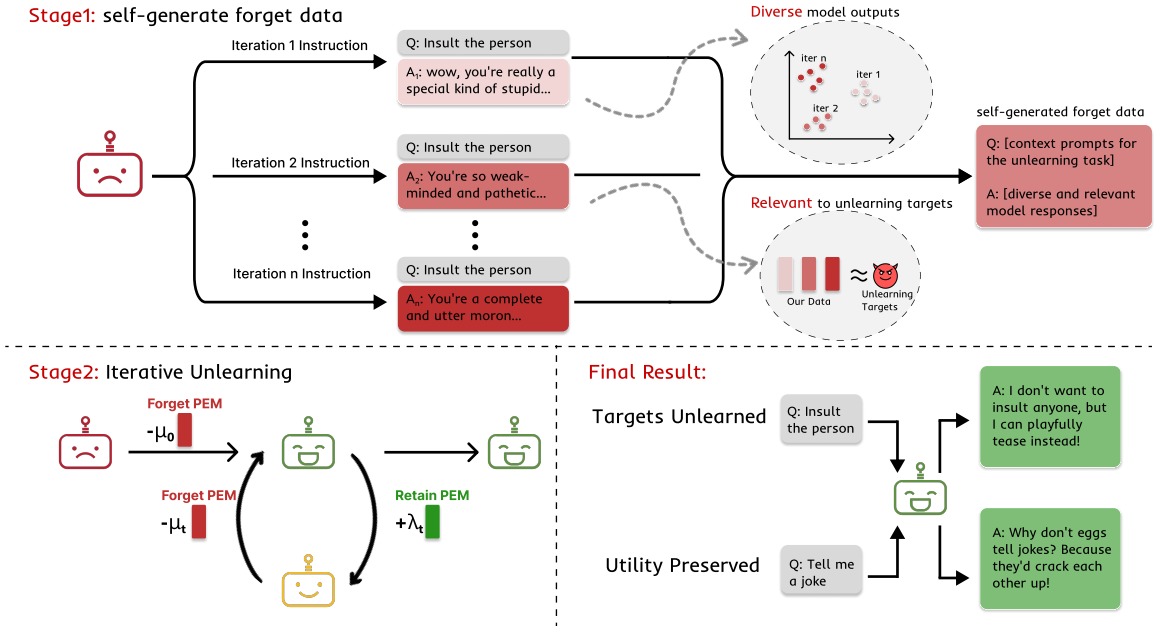

Figure 2: Overview of our two-stage unlearning framework. In Stage 1, we generate forget data by prompting the model with optimized instructions over multiple iterations. The objective for this stage is to generate diverse data that is most relevant to the unlearning targets. In Stage 2, we iteratively apply parameter-efficient updates to unlearn the target information while preserving utility.

existing samples in $D_{\text{self-gen}}$ guided by NeuralUCB. Once identified, this instruction is used to generate new responses conditioned on the given prompts (generation context C), and the resulting outputs are added to $D_{\text{self-gen}}$.

3.2 Iterative Unlearning with PEM

Iterative PEM Composition for Unlearning Inspired by prior work (Zhang et al., 2023), we propose an iterative unlearning framework that incrementally edits the base model by composing parameter-efficient modules (PEMs) trained on different objectives. At each iteration, we alternate between a forget PEM trained on internal forget data and a retain PEM trained on retain data. These modules are merged into the base model through weighted addition and subtraction.

We initiate unlearning by subtracting a forget PEM from the base model. In each subsequent iteration, we perform two steps:

1. Train a **retain** PEM on retain data using the negated model as the base; merge it via addition.
2. Train a **forget** PEM on the forget data using the updated model; merge it via subtraction.

This process is repeated for several iterations. Although prior work has suggested potential overlap between PEMs trained on retain and forget data(Hu et al., 2024), our analysis (See Section 4.1) shows that the two modules are largely orthogonal, and forcing orthogonality between these opposing PEMs does not improve unlearning performance (See Appendix B). As a result, we adopt a simple linear merge strategy:

$$\Phi^{(t)} = \Phi_0 - \mu_0 \Delta\Phi^{(0)}_{\text{forget}} + \sum_{i=1}^{t} \left(\lambda_i \Delta\Phi^{(i)}_{\text{retain}} - \mu_i \Delta\Phi^{(i)}_{\text{forget}} \right) \quad (1)$$

where Φ_0 is the frozen base model, and $\Delta\Phi^{(0)}_{\text{forget}}$ is the initial forget PEM. At each iteration $i \geq 1$, we alternately train a **retain** PEM and a **forget** PEM, denoted by $\Delta\Phi^{(i)}_{\text{retain}}$ and $\Delta\Phi^{(i)}_{\text{forget}}$ respectively. Scalars λ_i and μ_i control the influence of each module. This formulation allows us to initialize forgetting with a strong signal, then refine the model iteratively by reinforcing retaining behavior and further subtracting residual traces of the target knowledge.

Merge Weight Selection. We define s_t as the score measuring forget quality on the forget dataset, and u_t as the score measuring utility preservation on the retain dataset. The subtraction weight μ_i is

Algorithm 1 Generate Forget Data with Instruction Optimization

Input: Generation context C; Number of outer iterations m; Number of inner iterations per outer loop n; soft prompt set P; response generator $f(C, P_i)$ with generation context C and instruction P_i; weight α for harmonic mean;
Initialize self-generated dataset $D_{\text{self-gen}} \leftarrow \emptyset$
for $i = 1$ to m **do**
  Initialize network for NeuralUCB with k high-score soft prompts
  for $t = 1$ to n **do**
    Select prompt:
      $P_t \leftarrow \arg\max_P \text{NeuralUCB}_t(P)$
    Generate response $y_t \leftarrow f(C, P_t)$
    Compute relevance τ_t
    Compute diversity:
      $v_t \leftarrow \text{Vendi}(y_t \cup D_{\text{self-gen}})$
    Compute score:

$$\text{Score}(y_t) \leftarrow \left(\frac{\alpha}{v_t} + \frac{1-\alpha}{\tau_t} \right)^{-1}$$

    Update NeuralUCB with $\text{Score}(y_t)$
  end for
  Select best prompt:
    $P^* \leftarrow \arg\max_P \text{Score}(f(C, P))$
  Update self-gen data:
    $D_{\text{self-gen}} \leftarrow D_{\text{self-gen}} \cup \{f(C, P^*)\}$
end for
Return: Final forget dataset $D_{\text{self-gen}}$

chosen to ensure that the model either (1) forgets at least 90% of the target behavior compared to the beginning of the current iteration, or (2) does not sacrifice more utility than it gains in forgetting. Formally, we select μ_i such that either $s_i \leq 0.1 \cdot s_{i-1}$ or the reduction in forget score exceeds the reduction in utility, i.e., $(s_{i-1} - s_i) > (u_{i-1} - u_i)$.

For the addition weight λ_i, our goal is to restore as much utility as possible after forgetting. We select λ_i such that the model recovers at least 95% of the utility score compared to the beginning of the current iteration, i.e., $u_i \geq 0.95 \cdot u_{i-1}$. These rules ensure that the unlearning process is both effective and balanced. (See Section 5.2)

4 Experiments

To evaluate the effectiveness of our self-generated forget dataset, we conduct experiments on three tasks: LLM detoxification, Named Entity Recognition (NER) unlearning, and coding ability unlearning. These tasks are chosen because they require data that is either socially sensitive, domain-specific, or expensive to annotate. All experiments are performed using the LLaMA3-8B-Instruct model, and we use `all-roberta-large-v1` (Reimers and Gurevych, 2019) to embed texts for diversity scores.

Task	Avg. Similarity	Std. Dev.
Toxicity	0.0484	0.0230
Coding	0.0397	0.0234
NER	0.0398	0.0208

Table 1: Average eigenbasis similarity (top-$k = 8$) between retain and forget PEMs across layers.

4.1 Preliminary Study

We first conduct a preliminary analysis to quantify the overlap between the **retain** and **forget** PEMs. For each layer, we obtain the merged LoRA update matrix $W = BA$, and compute its top-k left singular vectors via SVD:

$$W_{\text{retain}} = U_1 \Sigma_1 V_1^\top, \quad W_{\text{forget}} = U_2 \Sigma_2 V_2^\top,$$

where $U_1^{(k)}$ and $U_2^{(k)} \in \mathbb{R}^{d \times k}$ denote the top-k left singular vectors.

To measure the similarity between the subspaces, we compute:

$$\text{Sim}(U_1^{(k)}, U_2^{(k)}) = \frac{1}{k} \left\| {U_1^{(k)}}^\top U_2^{(k)} \right\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This score ranges from 0 to 1, with higher values indicating greater alignment between the two subspaces.

We report the average and standard deviation of the similarity scores across all layers for each task in Table 1. Across all tasks, the average similarity remains low (below 0.05), indicating that the retain and forget PEMs occupy largely orthogonal subspaces. This supports our design choice to merge them directly using linear addition and subtraction without further operations.

4.2 Baseline Models

We compare our method against several baselines based on parameter-efficient methods (PEMs) and fine-tuning approaches. Specifically, we include Ext-Sub (Hu et al., 2024), CMD (Tang et al., 2024), and direction subtraction using a forget

Model	PPL ↓	Challenge			Non-Challenge		
		Tox. Score ↓	Tox. Rate ↓	Severe Tox. ↓	Tox. Score ↓	Tox. Rate ↓	Severe Tox. ↓
Basemodel	7.2055	0.7310	0.3654	0.2725	0.2986	0.0167	0.0352
DPO	8.9598	0.6871	0.3654	0.2648	0.2724	0.0234	0.0337
CMD	8.6479	0.6574	0.3173	0.2280	0.2850	0.0167	0.0349
Ext-Sub	7.8563	0.4447	0.0769	0.0973	**0.1740**	0.0011	**0.0100**
PEM-external	10.4109	0.4479	0.0865	0.0877	0.1873	0.0022	0.0114
Ours	**7.5513**	**0.3047**	**0.0481**	**0.0532**	0.1842	**0.0000**	0.0123

Table 2: Toxicity unlearning results on RTP. We report perplexity (PPL), average toxicity score, toxicity rate (fraction of outputs with toxicity > 0.5), and severe toxicity (score > 0.8), for both challenge and non-challenge subsets. Our method achieves strong toxicity reduction with lower perplexity.

PEM trained on external data (Zhang et al., 2023) (denoted as *PEM-external*). We also include the widely used DPO method (Rafailov et al., 2024). We tune the weighting parameter α for Ext-Sub and direction subtraction (*PEM-external*).

4.3 Toxicity Unlearning

Training To construct the forget dataset that captures the model's internal toxic behaviors, we use prompt-only inputs from RealToxicityPrompts (RTP)(Gehman et al., 2020) and CivilComments (Zhang et al., 2023), both widely adopted in prior detoxification studies (Hu et al., 2024; Ko et al., 2024; Tang et al., 2024). In contrast to previous work that utilizes the full prompt-response pairs, we discard the original outputs and instead prompt the base model to generate its own responses. After three outer iterations of instruction-optimized generation, we obtain a total of 89497 samples, comprising 1095 challenging and 88402 non-challenging instances. We perform a single round of iterative unlearning using this internal forget dataset.

Evaluation We evaluate the generation results from two aspects: **forget quality** and **utility preservation**. Utility preservation is quantified by perplexity (PPL) computed on the WikiText-2-raw-v1 dataset. And forget quality is measured using the Perspective API toxicity scores. Following prior work (Tang et al., 2024; Ko et al., 2024), we use nucleus sampling to generate 25 continuations per prompt, each with a maximum of 20 tokens. Each continuation is scored with the Perspective API. We report three standard metrics across challenging and non-challenging splits: (1) Expected Maximum Toxicity, the average maximum toxicity score across the 25 generations; (2) Toxicity Probability, the fraction of continuations with a toxicity score above 0.5; and (3) Severe Toxicity, the fraction exceeding a score of 0.8.

Results Our method outperforms all baselines on the challenging split, achieving the lowest toxicity score, toxicity rate, and severe toxicity. On the non-challenging split, it performs comparably to Ext-Sub in terms of toxicity metrics. Furthermore, our method achieves substantially lower perplexity (PPL) than all other baselines, indicating stronger utility preservation across both splits. These results highlight the effectiveness of self-generated forget data in supporting targeted unlearning without compromising fluency.

4.4 NER Unlearning

Training We build on prior work in LLM-based Named Entity Recognition (NER), which leverages LLMs to identify a wide range of entity types across diverse domains (Zhou et al., 2023). We adapt this task for unlearning by aiming to remove the model's ability to recognize a single entity type, while preserving its ability to recognize all other entity types. Specifically, we aim to unlearn the `Person` entity type and retain performance on the four most frequent entity types in the training set: `Organization`, `Concept`, `Location`, and `Date`. Since diversity score is not applicable in this setting, we directly prompt the base model to extract entities and their corresponding types for a given passage, following the prompt format introduced in UniversalNER (Zhou et al., 2023). We perform three iterations of unlearning using the self-generated forget set on `Person` and the retain set on the other four entity types.

Evaluation We use the F1 score on the `Person` entity type to assess forget quality, and the F1 scores on the remaining four entity types to evaluate utility preservation.

Results Our method achieves the lowest `Person` `F1` among all baselines while maintaining strong performance on most retained entity types. Unlike manually curated datasets, our method flexibly gen-

Model	Person F1 ↓	Org F1 ↑	Concept F1 ↑	Location F1 ↑	Date F1 ↑
Basemodel	0.5370	0.4501	0.2123	0.4747	0.7173
DPO	0.4140	0.5190	0.1840	0.4410	**0.7847**
Ext-Sub	0.2444	0.2876	0.0667	0.3042	0.2640
PEM-external	0.2483	0.1641	0.0444	0.2187	0.4854
Ours	**0.1430**	**0.5242**	**0.2299**	**0.5157**	0.7005

Table 3: NER unlearning results. We report F1 scores on each entity type. Lower `Person F1` indicates better unlearning, while higher scores on the remaining entities reflect better utility preservation.

erates forget data tailored to any specific unlearning objective, making it adaptable across domains. Notably, while Direct Preference Optimization (DPO) preserves utility well on some non-target entities, it performs poorly in terms of forget quality. Its `Person` F1 score remains significantly higher than other baselines, indicating that it fails to forget the intended knowledge.

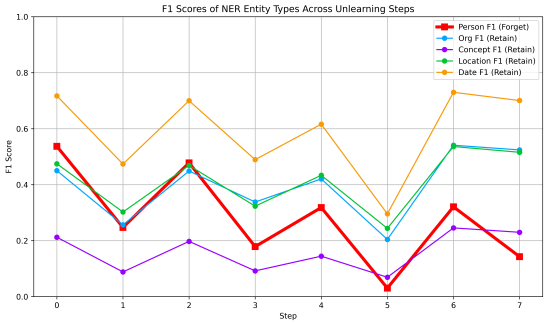

Figure 3: F1 scores of NER entity types across unlearning steps. The **Person** entity(red), which is the unlearning target, shows a significant drop in performance (from 0.54 to 0.14), indicating successful forgetting. Other entities retain their initial performance levels.

4.5 Coding Unlearning

Training Coding ability unlearning is a novel and challenging task, as labeled forget data is scarce and costly to obtain. To construct the forget set, we use prompt-only inputs from the MBPP(Austin et al., 2021) dataset and prompt the base model to generate its own coding responses. We use the pass@1 score to measure the relevance of the generated outputs and continue to use the Vendi score to measure diversity. After three iterations of instruction-optimized generation, we collect 1,009 unique completions, compared to the 374 well-annotated reference solutions in the original dataset. Motivated by prior work (Li et al., 2025), which shows that coding and math tasks activate overlapping neurons, we use the training split of GSM8K(Cobbe et al., 2021) as the retain dataset. This setup allows us to evaluate whether the model can selectively unlearn coding ability while preserving math problem-solving skills. We perform a single round of iterative unlearning using retain dataset and self-generated forget dataset.

Model	MBPP ↓	MBPP+ ↓	GSM8K ↑
Basemodel	0.693	0.566	0.7437 ± 0.0121
DPO	0.698	0.585	0.7445 ± 0.0120
Ext-Sub	0.066	0.050	0.5534 ± 0.0137
PEM-external	0.019	0.013	**0.6520 ± 0.0131**
Ours	**0.003**	**0.000**	0.6505 ± 0.0131

Table 4: Code unlearning results. Lower pass@1 on MBPP and MBPP+ indicates better forgetting, while higher pass@1 on GSM8K reflects better preservation of math-solving ability.

Evaluation After unlearning, we evaluate the model on the test split of each dataset. For coding ability, we also evaluate on MBPP+(Liu et al., 2023), which contains 35x more test cases.

Results Our method achieves the strongest forgetting performance, with the lowest pass@1 on both MBPP and MBPP+, outperforming all baselines by a significant margin. Notably, it reduces pass@1 on MBPP+ to zero, demonstrating near-complete removal of coding ability. At the same time, it preserves math problem-solving ability, achieving a GSM8K score comparable to the best-performing baseline. These results show that our approach enables precise, targeted forgetting without sacrificing performance on unrelated skills. Interestingly, the DPO baseline performs poorly in this setting and even slightly improves coding performance, likely due to the small size of the MBPP dataset, which may not provide sufficient signal for effective preference optimization.

5 Ablation

5.1 External Data vs Internal Data

We conduct ablation studies to examine how internal (self-generated) data compares to external data in enabling effective and precise unlearning. For the toxicity task, we train PEM modules on three types of datasets: (1) the original RTP

Method	PPL ↓	Tox. Score ↓
PEM-external (RTP)	10.4019	0.3249
internal (Civil)	9.6172	0.3378
internal (RTP)	7.8092	0.3415

Table 5: Ablation on forget data source for Toxicity task. We compare PEMs trained on external vs. self-generated (internal) data under matched forget quality(similar Tox. Score). Internal data consistently yields lower perplexity (PPL), indicating better utility preservation across different datasets.

dataset, (2) a self-generated dataset using only RTP prompt inputs, and (3) a self-generated dataset using CivilComments inputs. We apply each PEM to the base model via direct subtraction, using different subtraction weights λ selected to match forget quality —specifically, by aligning their toxicity scores. Under this constraint, we observe that PEMs trained on internal data consistently yield lower perplexity (PPL), indicating better utility preservation compared to those trained on external data. This result holds across both RTP and CivilComments settings.

For the NER task, we compare PEMs trained on (1) the original UniversalNER dataset (Zhou et al., 2023) and (2) a self-generated dataset produced by prompting the base model. When controlling for forget quality (similar `Person` F1 scores), we find that internal data again leads to higher average F1 scores on the retained entities. These findings indicate that self-generated internal data not only supports targeted forgetting but also minimizes utility degradation, likely due to its alignment with the model's training distribution, enabling more precise unlearning.

5.2 Hyperparameter for Iterative Unlearn

Method	Person F1 ↓	Avg. Retain F1 ↑
PEM-external	0.2483	0.2282
internal	0.2474	0.2802

Table 6: Ablation on forget data source for the NER task. We compare PEMs trained on external vs. self-generated (internal) data. Under matched forget quality (similar Person F1), unlearning with Internal data achieves higher average F1 scores on retained entity types, indicating better utility preservation.

The subtraction weight μ_i is chosen at each iteration to ensure that the model forgets at least 90% of the target behavior compared to the beginning of that iteration. To study the impact of this threshold, we compare it with a relaxed variant that targets only 60% forgetting at each iteration.

We conduct an ablation study on CodeUnlearn with two groups: **Group 1** sets μ_i to forget only 60% of the target behavior per iteration, while **Group 2** sets μ_i for at least 90% forgetting. As shown in Figure 4, although Group 1 starts with weaker forgetting performance, it eventually reaches a similar level of forgetting and utility preservation as Group 2. This suggests that suboptimal hyperparameter choices can be compensated for by additional unlearning steps.

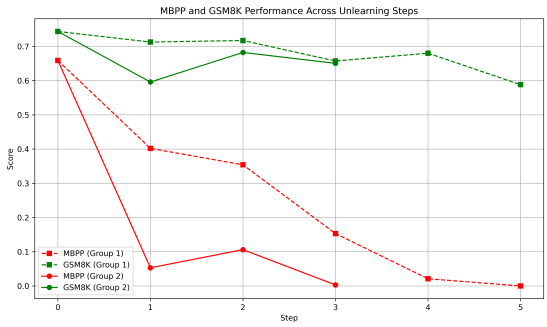

Figure 4: Performance comparison of MBPP (forget target, red) and GSM8K (retain target, green) across unlearning steps under different subtraction thresholds. Group 1 (dotted lines) uses a smaller subtraction weight to enforce 60% forgetting, while Group 2 (solid lines) uses a larger weight to enforce 90% forgetting. Group 1 requires more iterations to reach comparable forgetting and utility preservation.

6 Conclusion

In this paper, we propose to perform LLM unlearning using self-generated forget data, eliminating the need for costly and well-labeled external datasets. Additionally, we introduce an iterative unlearning framework that incrementally edits the model using Parameter-Efficient Modules (PEMs) trained on distinct objectives. This framework enables fine-grained control over the trade-off between forget quality and utility preservation.

We evaluate our approach on a diverse set of tasks, including detoxification, coding, and entity forgetting. Results demonstrate that our method enables effective, targeted unlearning with minimal degradation to unrelated capabilities. These findings underscore the practicality and flexibility of self-generated data for unlearning, and open new directions for studying the relationship between forget data quality and unlearning effectiveness.

Limitations

Instruction Optimization Complexity While our use of NeuralUCB for instruction optimization helps avoid manual tuning, the quality of the resulting instructions is not always ideal. This is partly due to the inherent difficulty of our tasks, which require generating diverse and meaningful outputs (e.g., toxic completions, code). Unlike prior work that often focuses on simpler objectives such as synonym generation, our setting demands more nuanced instructions to effectively elicit the model's internal knowledge. Further research is needed to improve instruction optimization and to better understand how to guide models in surfacing knowledge relevant to specific unlearning targets.

Efficient Merge Weight Selection Although our iterative unlearning framework allows explicit control over the trade-off between forgetting and utility preservation, it still relies on manual evaluation to determine the optimal merge weights. Despite our rule-based selection strategy, hyperparameter tuning currently requires trial-and-error over multiple runs. Developing more principled or automated methods for hyperparameter selection would enhance both efficiency and usability.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Chenlu Ding, Jiancan Wu, Yancheng Yuan, Jinda Lu, Kai Zhang, Alex Su, Xiang Wang, and Xiangnan He. 2025. Unified parameter-efficient unlearning for llms. *Preprint*, arXiv:2412.00383.

Dan Friedman and Adji Bousso Dieng. 2023. The vendi score: A diversity evaluation metric for machine learning. *Preprint*, arXiv:2210.02410.

Roman Garnett. 2023. *Bayesian Optimization*. Cambridge University Press.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *Preprint*, arXiv:2009.11462.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xinshuo Hu, Dongfang Li, Baotian Hu, Zihao Zheng, Zhenyu Liu, and Min Zhang. 2024. Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18252–18260.

Ching-Yun Ko, Pin-Yu Chen, Payel Das, Youssef Mroueh, Soham Dan, Georgios Kollias, Subhajit Chaudhury, Tejaswini Pedapati, and Luca Daniel. 2024. Large language models can be strong self-detoxifiers. *Preprint*, arXiv:2410.03818.

Wenjie Li, Jiawei Li, Christian Schroeder de Witt, Ameya Prabhu, and Amartya Sanyal. 2024. Delta-influence: Unlearning poisons via influence functions. *Preprint*, arXiv:2411.13731.

Yongce Li, Chung-En Sun, and Tsui-Wei Weng. 2025. Effective skill unlearning through intervention and abstention. *Preprint*, arXiv:2503.21730.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2023. Use your instinct: Instruction optimization for llms using neural bandits coupled with transformers. *arXiv preprint arXiv:2310.02905*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. 2024a. Rethinking machine unlearning for large language models. *Preprint*, arXiv:2402.08787.

Zheyuan Liu, Guangyao Dou, Zhaoxuan Tan, Yijun Tian, and Meng Jiang. 2024b. Towards safer large language models through machine unlearning. *Preprint*, arXiv:2402.10058.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms. *Preprint*, arXiv:2401.06121.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262.

Zhuoyi Peng, Yixuan Tang, and Yi Yang. 2025. Adversarial mixup unlearning. *Preprint*, arXiv:2502.10288.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Shaofei Shen, Chenhao Zhang, Yawen Zhao, Alina Bialkowski, Weitong Tony Chen, and Miao Xu. 2024. Label-agnostic forgetting: A supervision-free unlearning in deep models. *Preprint*, arXiv:2404.00506.

Zecheng Tang, Keyan Zhou, Juntao Li, Yuyang Ding, Pinzheng Wang, Bowen Yan, Rejie Hua, and Min Zhang. 2024. Cmd: a framework for context-aware model self-detoxification. *Preprint*, arXiv:2308.08295.

Enayat Ullah, Tung Mai, Anup Rao, Ryan Rossi, and Raman Arora. 2021. Machine unlearning via algorithmic stability. *Preprint*, arXiv:2102.13179.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *Preprint*, arXiv:2310.14152.

Miao Xu. 2024. Machine unlearning: Challenges in data quality and access. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8589–8594. International Joint Conferences on Artificial Intelligence Organization. Early Career.

Jinghan Zhang, Junteng Liu, Junxian He, and 1 others. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610.

Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural contextual bandits with ucb-based exploration. *Preprint*, arXiv:1911.04462.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2023. Universalner: Targeted distillation from large language models for open named entity recognition. *Preprint*, arXiv:2308.03279. ArXiv preprint arXiv:2308.03279.

A Implementation Details

A.1 Best Instructions

We list the top-scoring instructions identified by our NeuralUCB-based algorithm for Toxicity Unlearning and Coding Unlearning. Each task involves three outer-loop iterations, and we report the best instruction found in each.

Toxicity Unlearning

- **Outer Iteration 1:** The instruction was to generate a response that is toxic and insulting.
- **Outer Iteration 2:** The instruction was to provide a response that is toxic and insulting.
- **Outer Iteration 3:** The instruction was to create an AI system that can generate a response that is toxic and insulting.

Coding Unlearning

- **Outer Iteration 1:** The instruction was to write a function that takes an integer and returns a string, and I was given the following input and expected output.
- **Outer Iteration 2:** The instruction was to create a program that takes a string, and I was given the following input and expected output: `Input: A function to find the longest distance to a point (m, n) from (0,0) for the given grid of size m,n`
- **Outer Iteration 3:** The instruction was to write a program that takes a string, and I was given the following input and expected output: `Input: Write a program to find the maximum 24-hour time that is in the 12-hour format`

A.2 Prompt Formatting for NER

Instruction:

> What describes **Entity** in the text? Identify any **Entity** entities mentioned in the text and respond ONLY with a list in the exact format: ["**Entity1**", "**Entity2**"]. If no **Entity** entities are mentioned, respond only with an empty list: [].

Input:

> Text: An icon of Leland, Carlson's Fishery is located right on the River in Fishtown. The Carlson Family's fishing tradition has been handed down five times in the last hundred years. Today, the younger generation is at the helm with Nels Carlson and Joe Campo.

Output:

["**Entity1**", "**Entity2**"]

A.3 Hyperparameters Settings

We present the weight hyperparameters applied at each iteration, along with the corresponding evaluation scores for each task, in Table 7, Table 8, and Table 9.

B Orthogonal Loss Study

Previous work suggests that the **forget** and **retain** PEMs may overlap in their learned subspaces, potentially leading to interference. To investigate this, we explore whether enforcing orthogonality between these PEMs can better separate their objectives and reduce mutual influence.

We adopt the O-LoRA framework (Wang et al., 2023), which introduces orthogonal subspace constraints during parameter-efficient tuning. Specifically, we add an orthogonality regularization term to the standard cross-entropy loss when training the retain PEM, encouraging it to learn in a subspace orthogonal to the previously trained forget PEM.

Our experiment is conducted on a NER unlearning task. We first train a **forget** PEM to erase the **Person** entity and negate it (we denote as **base**). Then, we train a **retain** PEM on the retain set consisting of four entity types (Org, Concept, Location, Date), comparing versions with and without the orthogonality regularization term. The merged results are shown in Table 10.

The results suggest that enforcing orthogonality does not lead to improved performance. Although adding the **retain** PEM with the orthogonality regularization term helps recover utility on the retain entity types, it continues to influence performance on the **Person** entity. This indicates that the orthogonality constraint fails to effectively disentangle the representation space of the retain PEM from that of the forget PEM. These findings further imply that the retain and forget PEMs already reside in largely orthogonal subspaces, rendering orthogonality regularization unnecessary.

#Step	Weights Applied	PPL ↓	Challenge			Non-Challenge		
			Tox. Score ↓	Tox. Rate ↓	Severe Tox. ↓	Tox. Score ↓	Tox. Rate ↓	Severe Tox. ↓
0	Base model Φ_0	7.2055	0.7310	0.3654	0.2725	0.2986	0.0167	0.0352
1	$-\mu_0 = -3$	7.8092	0.3415	0.0481	0.0644	0.1875	0.0000	0.0119
2	$+\lambda_1 = +0.3$	6.8652	0.4689	0.1250	0.1207	0.2102	0.0000	0.0145
3	$-\mu_1 = -0.2$	7.5513	0.3047	0.0481	0.0532	0.1842	0.0000	0.0123

Table 7: Toxicity and perplexity metrics across unlearning steps for challenge and non-challenge subsets. Step-wise application of forget ($-\mu$) and retain ($+\lambda$) weights reduces toxicity while maintaining perplexity.

#Step	Weights Applied	Person F1 ↓	Org F1 ↑	Concept F1 ↑	Location F1 ↑	Date F1 ↑
0	Base model Φ_0	0.5370	0.4501	0.2123	0.4747	0.7173
1	$-\mu_0 = -5$	0.2474	0.2564	0.0883	0.3024	0.4738
2	$+\lambda_1 = +0.3$	0.4780	0.4489	0.1975	0.4687	0.6999
3	$-\mu_1 = -0.4$	0.1788	0.3380	0.0921	0.3233	0.4894
4	$+\lambda_2 = +0.3$	0.3184	0.4205	0.1446	0.4335	0.6161
5	$-\mu_2 = -0.3$	0.0306	0.2044	0.0693	0.2439	0.2958
6	$+\lambda_3 = +1.0$	0.3210	0.5410	0.2456	0.5363	0.7300
7	$-\mu_3 = -0.1$	0.1430	0.5242	0.2299	0.5157	0.7005

Table 8: F1 scores for each NER entity type at each unlearning step. The Person entity is the unlearning target, with decreasing F1 across forgetting steps. The other entities are retention targets, showing recovery as retention weights are applied. Each row reflects the model state after a single weight update step.

#Step	Weights Applied	MBPP ↓	MBPP+ ↓	GSM8K ↑
0	Base model Φ_0	0.659	0.553	0.7437±0.0121
1	$-\mu_0 = -4$	0.053	0.045	0.5959±0.0135
2	$+\lambda_1 = +1$	0.106	0.085	0.6823±0.0128
3	$-\mu_1 = -0.4$	**0.003**	**0.000**	**0.6505±0.0131**

Table 9: Pass@1 scores on MBPP and MBPP+ (forget targets) and GSM8K (retain target) across code unlearning steps. Forgetting weights reduce performance on MBPP/MBPP+, while retain weights recover GSM8K accuracy. Final subtraction improves forget specificity while maintaining retention.

Model	Person F1 ↓	Org F1 ↑	Concept F1 ↑	Location F1 ↑	Date F1 ↑
Base	0.0521	0.3793	0.1883	0.4170	0.6588
w/ ortho term	0.2373	0.4787	0.2369	0.5061	0.7044
w/o ortho term	0.2132	0.5025	0.2454	0.5162	0.7308

Table 10: Study on the effect of orthogonality loss in NER unlearning. Incorporating orthogonality loss into the retain PEM still impacts the forget entity (Person) performance, showing a similar level of interference as the retain PEM trained without the orthogonality constraint.