Decoupling Memories, Muting Neurons: Towards Practical Machine Unlearning for Large Language Models

Lishuai Hou, Zixiong Wang, Gaoyang Liu*, Chen Wang, Wei Liu, Kai Peng
Hubei Key Laboratory of Internet of Intelligence
School of EIC, Huazhong University of Science and Technology
{lishuaihou, zixwang, liugaoyang, chenwang, liuwei, pkhkust}@hust.edu.cn

Abstract

Machine Unlearning (MU) has emerged as a promising solution for removing the influence of data that an owner wishes to unlearn from Large Language Models (LLMs). However, existing MU methods, which require tuning the entire model parameters on the unlearned data with random labels or perturbed gradients, significantly degrade model utility, especially given the difficulty of accessing the original training data. This presents a key challenge: how can we achieve MU using only the unlearned data while preserving model utility? In this paper, we propose NeuMuter, a simple but effective MU method that eliminates the influence of unlearned data from LLMs by modulating the outputs of merely 1% of the neurons in the feed-forward network (FFN) modules within the Transformer blocks, minimizing disruption to the model's performance. We design a trainable masking scheme that decouples the memorization of different training data within the neurons of LLMs, allowing us to precisely identify and modify neurons associated with the unlearned data. Through comprehensive evaluations on two benchmarks across four different LLMs, we demonstrate that modifying the outputs of a few fraction of the total neurons can effectively achieve MU while preserving the model's utility across downstream tasks.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success, primarily due to their pretraining on vast amounts of text corpora, sourced from the internet (Chang et al., 2024b; Minaee et al., 2024; Chaves et al., 2024; Liu et al., 2023). When the training data contains unauthorized user information, practitioners are required to remove it from well-trained LLMs upon request, as individuals are granted the "Right to Be Forgotten" (RTBF) (Kurmanji et al., 2024) under regulations such as the EU's General Data Protection Regulation (GDPR) (Voigt and Von dem Bussche, 2017) and US's California Consumer Privacy Act (CCPA) (Bonta, 2022). To address this issue, Machine Unlearning (MU) has emerged as a promising paradigm, which aims to eliminate the effects of specific training samples from LLMs while preserving the remaining knowledge. Given the massive scale of LLMs, retraining from scratch is impractical, making the development of efficient unlearning methods for LLMs a significant challenge.

Most existing MU methods for LLMs require fine-tuning the entire model parameters by relabeling the unlearned data or reversing the contribution of gradient descent on the unlearned data (Jang et al., 2023; Wang et al., 2023; Chen and Yang, 2023; Yao et al., 2023; Eldan and Russinovich, 2023; Yao et al., 2024; Zhang et al., 2024a; Liu and Kalinli, 2024; Liu et al., 2024b; Lee et al., 2024; Maini et al., 2024). For example, MU methods has been proposed that focuses on solely leveraging the unlearned data. For example, Gradient Ascent methods (Jang et al., 2023; Zhang et al., 2024a) aim to maximize the loss of the unlearned data, whereas Data Relabeling (Eldan and Russinovich, 2023) seeks to minimize the loss of the relabeled unlearned data.

However, modifying the entire parameters of LLMs based solely on limited unlearned data often results in catastrophic forgetting, where removing specific data negatively impacts the model's utility (Yao et al., 2024; Fan et al., 2024). To mitigate this, other methods typically involve additional fine-tuning on the remaining training data (c.f. Table 1). For instance, Gradient Difference based methods (Yao et al., 2024; Lee et al., 2024; Maini et al., 2024) minimize the model's loss on the remaining data, while KL Minimization techniques (Wang et al., 2023; Chen and Yang, 2023; Liu and Kalinli, 2024; Yao et al., 2023; Liu et al., 2024b) minimize the Kullback-Leibler (KL) divergence between the predictions on remaining data of

[*]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 13978–13999
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

MU Methods	Remaining Training Set	Ratio of Trainable Parameters
Gradient Ascent	✘	100%
Data Relabeling	✘	100%
Gradient Difference	✓	100%
KL Minimization	✓	100%
NeuMuter	✘	**0.01%∼0.03%**

✓: Required ✘: Not-required

Table 1: Comparison with existing MU methods. Most existing methods either require modifying the whole parameters of LLMs, or the access to the remaining training set.

the original and the fine-tuned models. In practice, long-term storage of LLM training data is often infeasible due to high maintenance costs, extensive storage requirements, and privacy concerns (Cha et al., 2024). Consequently, these MU methods struggle to leverage the remaining training data to maintain model utility.

Therefore, a fundamental gap in MU for LLMs remains: how can we achieve MU using only the unlearned data while preserving model utility? To bridge this gap, we propose NeuMuter, a simple yet effective MU method that modifies the outputs of merely 1% of the neurons in the feed-forward network (FFN) modules within the Transformer blocks, which are associated with the data to be unlearned. The key insight behind NeuMuter is that specific neurons within the FFN modules play a crucial role in retaining the memory of training data (Geva et al., 2021, 2022). Instead of altering the entire model parameters, we selectively adjust only the neurons responsible for memorizing the unlearned data while preserving the vast majority of the network's parameters. This targeted intervention enables effective MU while avoiding severe utility degradation, eliminating the need for additional fine-tuning on the remaining training data.

To leverage this, we introduce a trainable neuron mask within FFN modules to decouple memorization and precisely identify neurons related to the unlearned data. Unlike existing neuron localization methods (Dai et al., 2022; Wu et al., 2023; Chang et al., 2024a; Chen et al., 2024), which primarily focus on identifying neurons linked to specific keywords, our approach targets the neurons most critical for unlearning entire sentences. Recognizing that different neurons contribute variably to the unlearned data, we utilize the mask values as scaling factors for the outputs of the selected neurons, rather than directly deactivating the selected neurons. By adaptively modulating the outputs of these neurons, we can effectively eliminate the model's responses related to the unlearned data while maintaining the overall utility of the LLMs.

Our contributions are summarized as follows:

- We propose NeuMuter, an MU method for LLMs, which modifies the intermediate outputs of only a few neurons associated with the unlearned data. Unlike existing approaches, NeuMuter can preserve the unlearned model utility without fine-tuning on the remaining training data.
- We propose a trainable mask scheme to decouple data memorization within LLM neurons, effectively identifying neurons specific to specific training data. Our method shows that data memorization is localized to a sparse subset of neurons, about one in hundred of the total neurons in FFNs of LLMs.
- We evaluate the performance of NeuMuter on two benchmarks across four LLMs. The experiment results show that NeuMuter outperforms five state-of-the-art MU methods in both MU performance and LLM utility. The code for NeuMuter has been released for reproducibility purposes[1].

2 Preliminary

2.1 Knowledge Neuron

In LLMs, Transformer block comprises multi-head self-attention (SelfAtt) modules and feed-forward network (FFN) modules, interconnected by residual connections (He et al., 2016). Let X^l denote the input of the l-th Transformer block, and the operations within these two modules can be formulated as follows:

$$Q_h^l = X^l W_h^{Q,l}, K_h^l = X^l W_h^{K,l}, V_h = X^l W_h^{V,l} \tag{1}$$

$$\text{Self-Att}_h^l(X^l) = \text{softmax}\left(Q_h^l (K_h^l)^T\right) V_h^l, \tag{2}$$

$$\text{FFN}^l(H^l) = \sigma\left(H^l (W_1^l)^T\right) W_2^l, \tag{3}$$

where Q_h^l, K_h^l , and V_h^l represent the sequences of query, key, and value vectors for the h-th attention

[1] https://github.com/SPHelixLab/NeuMuter

head of the l-th Transformer block, respectively. $W_h^{Q,l}$, $W_h^{K,l}$, $W_h^{V,l}$, W_1^l, and W_2^l are parameter matrices for the l-th Transformer block. H^l is the hidden state given by projecting the concatenation of all heads. σ is the activation function in the FFN module. For simplicity, we omit the scaling factor in self-attention and the bias terms.

As revealed by the previous work (Geva et al., 2021), the parameters of the neurons corresponding to the FFN's parameters W_1^l and W_2^l constitute the key-value memory pairs. Let $\mathbf{k}_i^l$ and $\mathbf{v}_i^l$ represent the parameters of the i-th neuron in the first and the second layer of the FFN, respectively. The neuron activation of the first layer $a_i^l = \sigma(\mathbf{k}_i^l \mathbf{h}^l)$ can be viewed as a memory coefficient to weight the second layer neuron with the parameters of $\mathbf{v}_i^l$ that is associated with a semantic or syntactic concept (Geva et al., 2022). The output hidden states of the FFN module of the l-th Transformer block can be rewritten as:

$$\mathbf{o}^l = W_2^l \mathbf{a}^l = \sum_{i=1}^{d} a_i^l \cdot \mathbf{v}_i^l, \tag{4}$$

where d is the dimension of the intermediate hidden states in FFN.

2.2 Unlearning Goal

In the realm of MU, the objective is to remove the effects of specific data from a trained model, so that the model's predictions after unlearning closely resemble those of a model that has been retrained from scratch on the remaining data. Given the n-th training sample consisting of t_n tokens, denoted as $\mathbf{x}_n = (w_1^n, w_2^n, \ldots, w_{t_n}^n)$, in the pretraining corpus $\mathcal{D}$, where $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the original generative LLM model $\mathcal{M}_o$ with L layers and parameters θ_o is usually trained using next-token prediction. The model characterizes the conditional probability based on given prompts: $P_{\theta_o}(w_t \mid w_{<t})$. Formally, let $\mathcal{D}_f \subseteq \mathcal{D}$ represent the subset of training data we intend to unlearn, and $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$ denote the subset of data that remains. Our goal is to make an original model $\mathcal{M}_o$ with parameters θ_o to forget $\mathcal{D}_f$ and obtain an unlearned model $\mathcal{M}_u$ with parameters θ_u that contains minimal influence from $\mathcal{D}_f$ while maintaining the model's performance on $\mathcal{D}_r$. Since obtaining a retrained model of LLMs is nearly impossible, we base our work on prior researches (Kurmanji et al., 2024; Chen and Yang, 2023; Gao et al., 2024) and concentrate on the relationship between the prediction distributions of the original model and the model after unlearning, for both the unlearned and the remaining data.

Thus, the aim of MU in the context of LLMs is twofold: (i) Ensuring effective unlearning of the unlearned data; (ii) Preserving the model's effectiveness on the remaining data. Therefore, we formulate the objective of MU for LLMs as:

$$\arg\max_{\theta_u} \mathbf{dist}(\mathcal{M}_u(\mathcal{D}_f); \mathcal{M}_o(\mathcal{D}_f)), \tag{5}$$

$$\arg\min_{\theta_u} \mathbf{dist}(\mathcal{M}_u(\mathcal{D}_r); \mathcal{M}_o(\mathcal{D}_r)), \tag{6}$$

where $\mathbf{dist}$ represents the metric used to measure the divergence between the predictions of the unlearned model $\mathcal{M}_u$ and the original model $\mathcal{M}_o$, such as KL divergence (Kurmanji et al., 2024; Chen and Yang, 2023) and cross-entropy loss (Zhang et al., 2024a; Yao et al., 2024).

3 Design of NeuMuter

NeuMuter modifies the intermediate outputs of a small subset of LLM neurons associated with the unlearned data, therefore it consists of two main phases (c.f. Figure 1): Memorization Neuron Localization and Memorization Removal.

3.1 Memorization Neuron Localization

Several neuron localization methods exist for LLMs (Dai et al., 2022; Wu et al., 2023; Chang et al., 2024a; Chen et al., 2024), which primarily target predetermined information within a sentence, such as tail entities (Dai et al., 2022) or personally identifiable information (PII) (Wu et al., 2023; Chen et al., 2024). Directly applying existing localization methods to our MU approach would only select neurons associated with specific phases or words in the sample, thereby neglecting other relevant information in the unlearned data and result in incomplete unlearning.

In NeuMuter, we aim to find the smallest subset of knowledge neurons responsible for the unlearned data prediction, as these neurons directly impact the model's loss on the unlearned data. To achieve this goal, we introduce a trainable mask matrix $\mathbf{M} \in \mathbb{R}^{L \times d}$ into the FFNs within the Transformer blocks of LLMs. Specifically, each learnable mask vector $\mathbf{m} \in \mathbb{R}^d$ is inserted between the two linear layers of each FFN module. Formally,

$$\mathbf{o}_u^l = W_2^l \mathbf{a}^l \odot \mathbf{m}^l = \sum_{i=1}^{d} m_i^l \cdot a_i^l \cdot \mathbf{v}_i^l, \tag{7}$$

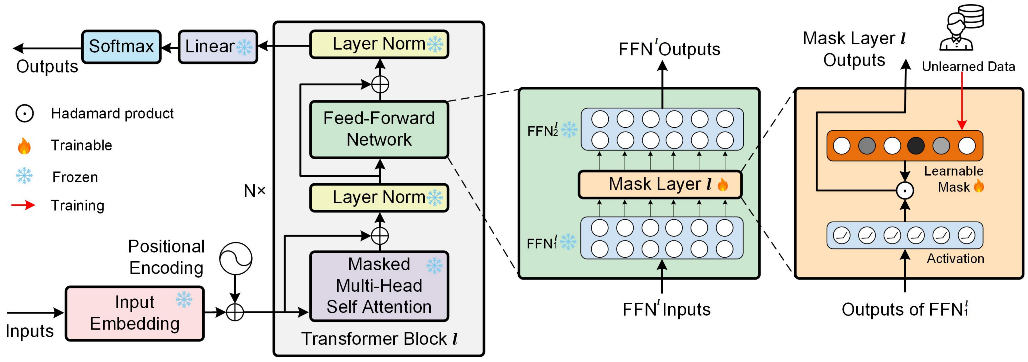


Figure 1: Illustration of NeuMuter. The shading of parameters in the mask indicates the sensitivity of the corresponding knowledge neurons to the unlearned data; the darker the color, the more critical that knowledge neuron is for the model's memory of the unlearned data.

where $\mathbf{m}^l$ is the mask vector of $\mathbf{M}$ applied to the l-th FFN module, m_i^l is the i-th value of $\mathbf{m}^l$ corresponding to the i-th neuron in the second layer of the FFN, and $\odot$ denotes the Hadamard product. In NeuMuter, we train the mask by increasing the prediction loss on $\mathcal{D}_f$, allowing us to identify which neurons are relevant to the unlearned content:

$$\mathcal{L}_f(\mathcal{D}_f;\theta) = -\sum_{\mathbf{x}_j \in \mathcal{D}_f} \sum_{t=1}^{t_j} \log P_\theta(w_t^j | w_{<t}^j), \quad (8)$$

where θ represents the combination of θ_o and the added mask $\mathbf{M}$. P denotes the probability of the t-th word w_t^j occurring, given the preceding $t-1$ words $w_{<t}^j$ in the input sequence.

However, the neurons selected may include not only those related to the unlearned data but also some that encode general knowledge, such as grammar and syntax, referred to as "universal neurons" (Gurnee et al., 2024). Modifying all these neurons would significantly impair the model's performance. To filter out these neurons, we use a Transformer-based masked language model (MLM), such as BERT, to generate neighbor samples for $\mathcal{D}_f$ (Bărbulescu and Triantafillou, 2024; Mattern et al., 2023a), which share the general knowledge but differ in specific information of $\mathcal{D}_f$. We generate K neighbor samples for each unlearned sample $\mathbf{x} \in \mathcal{D}_f$ to form $\tilde{\mathcal{D}}_n$ by replacing a randomly selected proportion of tokens in $\mathbf{x}$ with replacement ratio r.

Due to the absence of knowledge regarding the unlearned data in the MLM , maintaining the model's performance on these neighbor samples can help filter out universal neurons, leaving only those neurons specific to $\mathcal{D}_f$. Therefore, we train the mask by minimizing the prediction KL divergence between the LLM's outputs on the neighbor samples before and after applying the mask:

$$\mathcal{L}_n(\tilde{\mathcal{D}}_n;\theta) = \sum_{\tilde{\mathbf{x}}_k \in \tilde{\mathcal{D}}_n} \sum_{t=1}^{|\tilde{\mathbf{x}}_k|} \mathrm{KL}\big(M_{\theta_o}(\tilde{w}_t^k | \tilde{w}_{<t}^k) \quad (9)$$
$$\| M_\theta(\tilde{w}_t^k | \tilde{w}_{<t}^k)\big).$$

To further improve the precision of neuron localization, we enhance the sparsity to our mask, focusing on selecting only the most critical neurons related to the unlearned data. We assume each mask value m_i^l to be an independent random variable that follows a hard concrete distribution HardConcrete$(\log \alpha_i^l, \beta_i^l)$ (Louizos et al., 2018) with temperature β and location α_i^l to improve the robustness of mask training:

$$s_i^l = \sigma\Big(\frac{1}{\beta}\big(\log \frac{\mu_i^l}{1-\mu_i^l} + \log \alpha_i^l\big)\Big), \quad (10)$$

$$m_i^l = \min(1, \max(0, s_i^l(\zeta - \gamma) + \gamma)), \quad (11)$$

$$R(\mathbf{M}) = 1 - \frac{1}{N_m} \sum_{l=1}^{L} \sum_{i=1}^{d} \sigma(\log \alpha_i^l - \beta \log \frac{-\gamma}{\zeta}), \quad (12)$$

where σ denotes the sigmoid function, γ and ζ are constants, μ_i^l is randomly sampled from uniform distribution $\mathcal{U}(0,1)$, and $N_m = L \cdot d$. Instead of directly training m_i^l, the random variable s_i^l parameterized by α_i^l is learned to approximate a discrete Bernoulli distribution (Maddison et al., 2022). We restrict the values in the mask to the range of 0 to 1 as shown in Eq. 11. $R(\mathbf{M})$ is the L_0 regularization to minimize the number of localized neurons and allows the mask values to measure the contribution of neurons to the memorized data. The smaller the value, the greater the memory sensitivity of the neuron. Finally, we train the mask by minimizing the following loss function:

$$\mathcal{L}_{MU} = -\mathcal{L}_f(\mathcal{D}_f;\theta) + \lambda\mathcal{L}_n(\tilde{\mathcal{D}}_n;\theta) + \eta R(\mathbf{M}), \tag{13}$$

where λ and η are hyper-parameters. Throughout the training process, all weights of the LLMs remain frozen, with only $\mathbf{M}$ being trainable.

After training $\mathbf{M}$, we select neurons with mask values below a threshold ϵ as knowledge neurons $\mathcal{N}$. To determine ϵ, we sort all values in $\mathbf{M}$ in ascending order. Then, we compute the total energy of these elements, where energy is defined as the absolute value of each element. After sorting the elements, we incrementally accumulate the energy until the cumulative energy does not exceed 5% of the total energy. The value corresponding to this point is selected as the threshold ϵ.

3.2 Memorization Removal

Having identified the neurons $\mathcal{N}$ associated with $\mathcal{D}_f$, a straightforward approach to unlearn is to deactivate the identified neurons. However, complete deactivation of these neurons may compromise the model's utility, evidenced by their non-zero activations for unrelated inputs (Foster et al., 2024).

Referring back to Eq. 4 in Section 2.1, $\mathbf{a}^l$ can be seen as the dynamic coefficients of $\mathbf{v}^l$, where increasing (or decreasing) the coefficient can enhance (or diminish) the probability of the concepts contained in the corresponding value vector in the model's output distribution (Geva et al., 2022). Thus, the values of the trained mask can therefore be regarded as indicators of the level of memorization of the unlearned data for the neurons in $\mathcal{N}$.

We apply the values of the mask $\mathbf{M}$ as reduction factors to the selected neurons to reduce the model's memorization of the unlearned data while keeping the other neurons unchanged:

$$m_i^l = \mathbb{1}_{\{\mathbf{v}_i^l \in \mathcal{N}\}} \cdot m_i^l + \mathbb{1}_{\{\mathbf{v}_i^l \notin \mathcal{N}\}}, \tag{14}$$

where $\mathbb{1}_{\{\cdot\}}$ is indicator function, yielding a value of 1 if the condition within $\{\cdot\}$ is met and 0 otherwise. The pseudocode is elaborated in Algorithm 1.

4 Performance Evaluation

4.1 Experiment Setup

Models and Datasets. We follow the settings in existing MU works (Jang et al., 2023; Bărbulescu and Triantafillou, 2024) to perform a fair comparison with them. We use the GPT-Neo model family (125M, 1.3B, 2.7B) (Black et al., 2021), and the unlearned samples are selected randomly from the Training Data Extraction Challenge[2]. We also use the TOFU benchmark (Maini et al., 2024) to further evaluate the performance of our NeuMuter (5).

Comparison methods. We compare the performance of our NeuMuter with five state-of-the-art MU methods targeted at LLMs.

- *Gradient Ascent (GA)* (Jang et al., 2023). This method achieves unlearning by reversing the original training loss on the unlearned data, thereby removing the contribution of the unlearned data to model training.

- *Selective Gradient Ascent (SGA)* (Bărbulescu and Triantafillou, 2024). This method refines GA by selectively applying gradient ascent to samples with memory accuracy above a specified threshold in the unlearned data. It dynamically adjusts the target set during training, focusing on high-memory samples until overall unlearning criteria are met.

- *Task Arithmetic (TA)* (Ilharco et al., 2023). This method fine-tunes the original LLM on the unlearned data to identify the direction in which the relevant parameters are updated during the learning process. The updated weights are then subtracted from the original model, thereby removing the information learned from the unlearned data.

- *Deliberate Imagination (DI)* (Dong et al., 2024). This method leverages self-distillation to reduce the sampling probability of unlearned tokens while increasing the probability for others. By modifying the logits of the student model, it selectively unlearns targeted data while retaining overall model performance.

- *Integrated Gradients (IG)* (Chang et al., 2024a; Dai et al., 2022; Wu et al., 2023). This method identifies FFN neurons associated with specific tokens (e.g., names or phone numbers) by computing the accumulated gradients from a zero vector to the original hidden state in the second FFN layer. Unlearning is

[2] https://github.com/google-research/lm-extraction-benchmark

Model	Method	Unlearned Data		Downstream Tasks			
		EL$_{10}$ (%) ↓	**MA(%)** ↓	**ACC(%)** ↑	**F1(%)** ↑	**CE** ↓	**PPL** ↓
GPT-Neo (125M)	Original	33.81 (±0.24)	78.64 (±1.12)	43.09 (±2.00)	9.94 (±0.25)	3.77 (±0.14)	31.00 (±1.80)
	GA	0.13 (±0.27)	29.43 (±1.93)	41.43 (±2.78)	1.42 (±0.98)	6.77 (±1.12)	638.23 (±112.47)
	TA	0.20 (±0.22)	29.73 (±0.93)	36.06 (±1.50)	5.64 (±0.80)	4.81 (±0.30)	120.15 (±19.83)
	SGA	**0.04** (±0.12)	29.37 (±1.10)	40.93 (±1.25)	1.32 (±0.50)	6.96 (±0.80)	803.82 (±137.92)
	DI	0.10 (±0.08)	29.33 (±0.25)	42.67 (±0.50)	9.83 (±0.50)	3.88 (±0.15)	39.90 (±5.67)
	IG	1.01 (±0.17)	29.11 (±1.30)	40.50 (±1.50)	5.55 (±0.60)	5.10 (±0.20)	200.85 (±28.43)
	NeuMuter	2.62 (±0.57)	**28.63** (±0.22)	**42.67** (±0.25)	**10.23** (±0.27)	**3.83** (±0.25)	**39.17** (±3.12)
GPT-Neo (1.3B)	Original	66.45 (±1.12)	91.22 (±1.24)	50.01 (±1.60)	12.26 (±0.20)	3.25 (±0.20)	16.30 (±1.55)
	GA	2.12 (±0.74)	31.01 (±1.77)	48.38 (±1.39)	11.47 (±0.61)	3.36 (±0.31)	7103222.30 (±7654.32)
	TA	3.11 (±0.50)	31.74 (±1.20)	44.21 (±1.00)	7.11 (±0.70)	4.19 (±0.30)	47.20 (±6.14)
	SGA	4.23 (±0.60)	30.89 (±1.30)	46.98 (±1.51)	11.85 (±0.59)	3.30 (±0.29)	50.91 (±8.62)
	DI	**0.15** (±0.10)	31.11 (±0.55)	48.37 (±0.66)	9.66 (±0.40)	3.30 (±0.25)	20.26 (±2.73)
	IG	1.90 (±0.30)	31.97 (±1.50)	36.59 (±1.52)	9.24 (±0.50)	5.68 (±0.27)	246.38 (±33.61)
	NeuMuter	2.89 (±0.45)	**30.89** (±0.50)	**49.87** (±0.40)	**12.10** (±0.23)	**3.26** (±0.24)	**17.92** (±2.91)
GPT-Neo (2.7B)	Original	70.20 (±0.64)	93.22 (±0.75)	52.58 (±1.80)	12.33 (±0.25)	3.25 (±0.15)	14.34 (±0.85)
	GA	**0.27** (±0.44)	33.04 (±1.45)	50.21 (±0.64)	11.45 (±0.52)	3.78 (±0.35)	5231639.15 (±4321.65)
	TA	0.55 (±0.10)	33.10 (±1.00)	49.58 (±1.20)	8.90 (±0.50)	3.74 (±0.20)	24.15 (±3.89)
	SGA	1.25 (±0.25)	32.85 (±0.71)	51.40 (±1.20)	11.56 (±0.40)	3.55 (±0.25)	298.52 (±48.19)
	DI	0.30 (±0.05)	33.33 (±0.56)	51.91 (±0.25)	12.05 (±0.50)	3.25 (±0.15)	16.88 (±1.94)
	IG	1.74 (±0.20)	33.83 (±1.05)	37.62 (±1.10)	7.84 (±0.40)	6.01 (±0.30)	341.15 (±39.22)
	NeuMuter	3.77 (±0.54)	**32.79** (±0.35)	**52.48** (±0.21)	**12.39** (±0.14)	**3.14** (±0.01)	**16.52** (±2.27)

Table 2: Performance comparison of NeuMuter with five baselines. **Bold** results indicate the best performance.

then achieved by zeroing out the activations of these neurons.

Evaluation Metrics. We evaluate NeuMuter in terms of both unlearning effectiveness and model utility for the unlearned LLMs.

From the effectiveness perspective, we use memorization accuracy (MA) and extraction likelihood (EL) (Jang et al., 2023) to quantify the memorization of LLMs with respect to the given unlearned data. Additionally, we perform a loss-based membership inference attack (MIA) on the unlearned LLMs to assess how much information about the unlearned data remains in the model, following the approach in (Bărbulescu and Triantafillou, 2024).

For the utility, we quantify the performance of unlearned LLMs using various metrics, including classification accuracy (ACC) on NLP classification tasks, F1 score (F1) and cross-entropy loss (CE) on dialogue reasoning benchmarks, and perplexity (PPL) on validation corpus. The detailed description of the evaluation metrics and experimental details is available in Appendix B.

4.2 Performance of NeuMuter

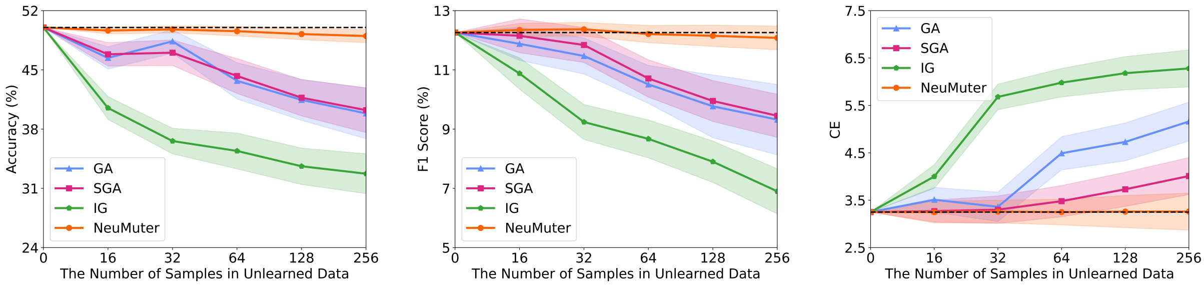
We evaluate NeuMuter alongside five comparison methods on GPT-Neo models, with results shown in Table 2. Average metrics across tasks and datasets are reported, with detailed results in Appendix C. NeuMuter outperforms other methods in unlearning with minimal utility loss, particularly for smaller models. It reduces text classification accuracy by only 0.42%, while improving F1 scores by 0.29% in dialogue tasks. In contrast, GA, SGA, and TA cause significant performance drops, especially in generation tasks. Existing MU methods struggle with small models, making NeuMuter more suitable for resource-constrained hardware.

As model size increases, NeuMuter remains effective. For the 2.7B GPT-Neo model, it only reduces classification accuracy by 0.1%, maintaining original F1 scores. GA and SGA cause nearly a 7% drop in F1, while DI matches NeuMuter in classification but lags in dialogue tasks.

NeuMuter generates more fluent outputs, with lower CE and PPL compared to other methods. GA has the worst fluency because gradient ascent approach softens the probability distribution. NeuMuter's minimal impact on utility comes from modifying only the neurons related to $\mathcal{D}_f$. Although NeuMuter may show slightly higher EL values, this isn't a disadvantage, as GA and SGA often over-forget, generating repetitive content. NeuMuter, by targeting critical neurons and filtering out universal ones, strikes a better privacy-utility balance.

4.3 Impact of Unlearned Data Size

To investigate the impact of the unlearned data size on the performance of NeuMuter, we set the number of $\mathcal{D}_f$ to 16, 32, 64, 128, and 256, using the

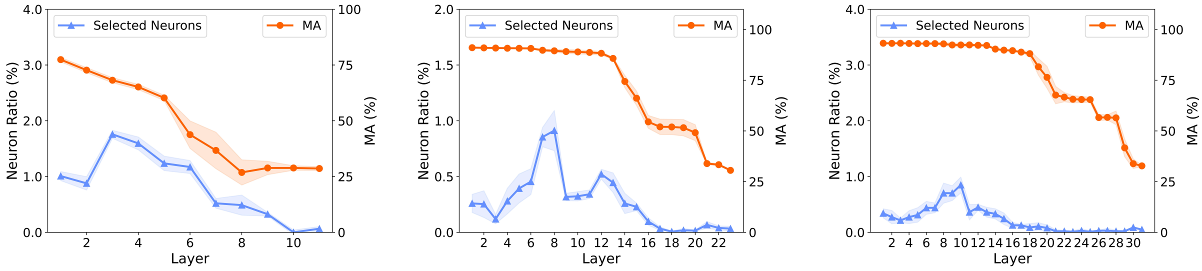


(a) Acc. on Classification Datasets

(b) F1 Score on Dialogue Datasets

(c) CE on Dialogue Datasets

Figure 2: The impact of the size of unlearned dataset on the performance of NeuMuter and comparisons.



(a) On GPT NEO 125M.

(b) On GPT NEO 1.3B.

(c) On GPT NEO 2.7B.

Figure 3: The impact of the neuron masks added to different Transformer blocks.

GPT-Neo model with 1.3B parameters. The experimental results are shown in Figure 2. NeuMuter outperforms other methods across model sizes and remains robust as the number of unlearned samples increases. For 256 unlearned samples, it achieves 49.00% accuracy in text classification with only a 1.01% drop from the original model. With fewer samples, NeuMuter even outperforms the original in language generation. Performance slightly declines when unlearned samples exceed 32 due to more selected neurons.

In contrast, gradient-ascent-based methods suffer significant utility loss as unlearned data grows. For fewer samples ($s = 16$), SGA performs well, with only a 0.11% drop, but with $s = 256$, it causes a 22.92% loss in generation ability. IG performs the worst by indiscriminately deactivating neurons, resulting in large performance degradation. Increased error margins are observed as the number of unlearned samples rises, likely due to variations in the order and content of unlearned data.

4.4 Impact of Masks in Transformer Blocks

In NeuMuter, we by default add masks to all Transformer blocks. Two questions naturally arise: Are all Transformer blocks are equally important for unlearning $\mathcal{D}_f$? If not, can MU be achieved by adding masks only to the most important blocks? To address these questions, we first add masks to all layers and run NeuMuter to unlearn 32 samples. Then, based on the trained masks, we respectively select the masked neurons in the first $l \in [1, \ldots, L]$ layers and measure the model's MA on $\mathcal{D}_f$. Additionally, we calculate the ratio of neurons selected in each layer to the total number of neurons therein, based on the masks.

Figure 3 shows that the blocks critical for reducing the model's memory of unlearned data are generally in the middle and later parts of the LLMs, even though NeuMuter-trained masks select more neurons in the early blocks. This effect becomes more pronounced as model size increases. For the 1.3B model, masking the first 10 blocks results in a 2.26% reduction in MA, while masking later blocks leads to a 55.03% drop. In the 2.7B model, adding masks to the first 14 blocks has minimal impact on MA. This is because the early blocks mainly process input prompts, while later blocks are more involved in reasoning and token prediction.

Additionally, focusing on critical blocks identified in Figure 3 and applying masks only to those blocks yields similar or better results than masking all layers shown in Table 3. In 1.3B model, modifying 8 layers instead of all blocks results in a 0.16% lower MA and a 0.16% improvement in ACC, indicating that selective modification can improve

Model	Masked Blocks	MA (%)	ACC (%)	F1 (%)	CE	PPL
GPT-Neo (125M)	All (1-12)	28.63	42.67	**10.23**	3.83	39.17
	2-8	**28.60**	**42.79**	10.16	**3.79**	**35.78**
GPT-Neo (1.3B)	All (1-24)	30.89	49.87	12.05	3.26	**17.92**
	10-16, 21	**30.73**	**50.03**	**12.10**	**3.25**	18.08
GPT-Neo (2.7B)	All (1-32)	32.79	52.40	**12.39**	**3.14**	**16.52**
	14-21, 25-26, 28-30	**32.40**	**52.48**	12.26	3.16	16.68

Table 3: Performance comparison when inserting masks in different Transformer blocks within the LLMs.

unlearning while preserving model performance.

4.5 Performance from Privacy Perspective

To assess whether unlearned LLMs truly eliminate the memorization of unlearned data, we adopt MIAs (Mattern et al., 2023b; Niu et al., 2024) to evaluate the performance of NeuMuter from a privacy perspective. In our experiment, we conduct a loss-based MIA in LLMs (Bărbulescu and Triantafillou, 2024). More details about the MIA are provided in the Appendix B.3. Figure 4 shows the probability density curves of the membership inference attack (MIA). NeuMuter consistently offers strong privacy guarantees across models of different sizes, with significantly lower loss distances to validation data after unlearning compared to the original model. In the GPT-Neo (1.3B) model, NeuMuter shows 19 unlearned samples with loss distances in the range $[0, 0.5]$, compared to 15 for SGA and 7 for GA. Although IG performs better than NeuMuter in the GPT-Neo (2.7B) model, it severely harms model utility, as shown in Table 2.

Furthermore, we observe that NeuMuter and GA show long-taileddistributions, with more samples exceeding higher thresholds than SGA or IG. In the GPT-Neo (1.3B) model, NeuMuter and GA have 4 and 15 samples with loss distances over 1.00, respectively, while SGA only has 2. This is because SGA treats each sequence individually during unlearning. GA, however, sometimes leads to over unlearning, as seen in the GPT-Neo (2.7B) model, where 4 samples have loss distances greater than the maximum observed in the original model.

5 Evaluation on TOFU Benchmark

To further evaluate NeuMuter, we utilize the recently introduced TOFU benchmark and compare it against the MU methods included in the benchmark. More details are shown in Appendix D.

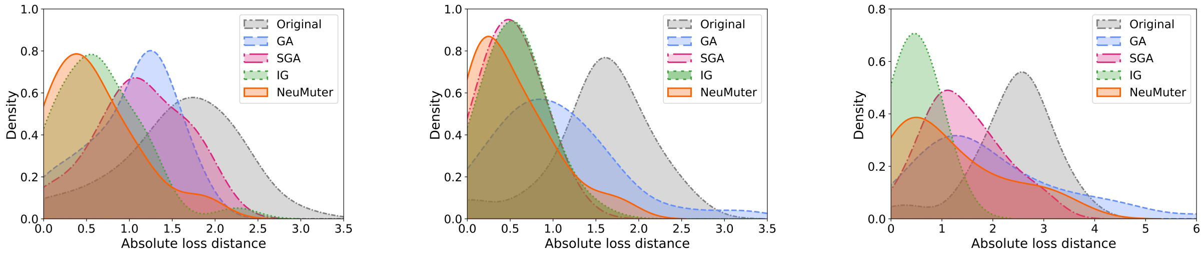
Figure 5 presents the unlearning results in terms of forget quality and model utility for synthetic unlearned profiles at 1%, 5%, and 10%. The results show that NeuMuter outperforms all baselines across different scenarios, closely matching the MU performance of the retrained model. Our findings demonstrate that effective MU can be achieved without compromising model utility and without relying on the remaining training set. It can preserve model utility, especially when the unlearned set is larger (e.g., 5% or 10%). NeuMuter also achieves higher p-values in forget quality, indicating better alignment with the retrained model. For the 1% unlearned set, all baselines show near-vertical trajectories, suggesting easier unlearning. However, their forget quality is still lower than NeuMuter's. As the unlearned data increases, unlearning trajectories become unstable. With 5% unlearned data, GA and KL improve forget quality but at a significant cost to utility, making them difficult to fine-tune. At 10%, GD achieves forget quality similar to NeuMuter but struggles with utility. Overall, methods using the retained set (e.g., GA) perform better than those focusing on minimizing unlearned set loss.

6 Conclusion

In this paper, we introduce NeuMuter, a practical MU method that selectively modifies the intermediate outputs of merely 1% of the neurons in FFN modules of LLMs. NeuMuter employs a trainable masking mechanism within the FFN modules to accurately identify neurons that store the memorization of unlearned data. By modulating the outputs of these neurons, NeuMuter effectively removes undesired information while preserving model utility—without requiring tuning the entire parameters of LLMs on the remaining training data. Extensive evaluation on nine datasets across four LLMs demonstrates that NeuMuter not only achieves superior MU performance but also preserves model utility compared to existing MU methods.

Limitations

Despite the effectiveness of NeuMuter in removing the memorization of the training data in LLMs, our current study has several limitations. First, we follow previous work (Geva et al., 2021, 2022) and assume that FFN modules within the Transformer

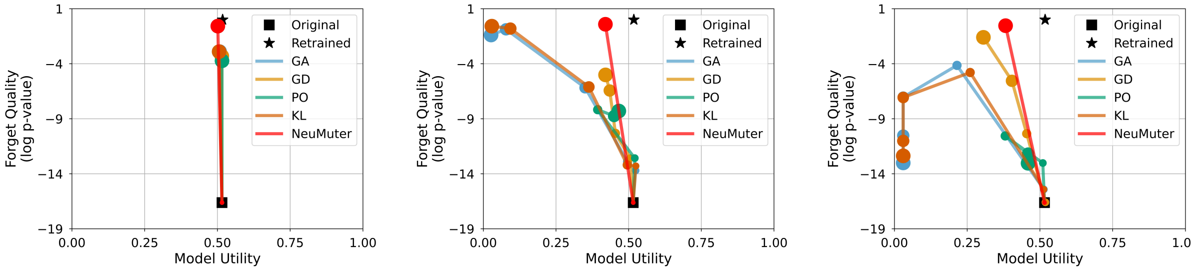


(a) MIAs on GPT-Neo (125M).

(b) MIAs on GPT-Neo (1.3B).

(c) MIAs on GPT-Neo (2.7B).

Figure 4: The results of MIA for NeuMuter and the comparison methodson on 32 unlearned samples.



(a) Unlearned Data Size 1%.

(b) Unlearned Data Size 5%.

(c) Unlearned Data Size 10%.

Figure 5: Performance of NeuMuter on TOFU benchmark with respect to Phi-1.3B model. The size of the markers corresponds to the number of epochs completed by different methods for MU.

blocks are the primary components of LLMs responsible for memorizing the information of training data. As a consequence, our method is focused solely on the localization of knowledge neurons within FFNs, without considering other critical model components, such as self-attention modules. Future research exploring how other model modules contribute to the memorization of training data could further enhance our MU method and provide a more comprehensive understanding of MU process.

Second, while our method reduces the risk of verbatim recall of training data, it relies on a limited set of evaluation metrics. Incorporating more diverse evaluation metrics, particularly those focusing on interpretability and model behavior, would offer a more holistic assessment of our approach's effectiveness. Such metrics could help identify potential weaknesses in our method that are not fully captured by the current evaluation framework of MU.

Additionally, due to computational constraints and the Source unknowability of LLM training data, we have not conducted experiments on very large-scale models such as LLaMA2 and LLaMA3. Future work could address the scalability of our unlearning algorithm, testing it on larger models and exploring its applicability to a broader range of unlearning tasks and benchmarks. This would help establish the generalizability and robustness of our approach across different model sizes and domains.

Ethics Statement

In an era where large amounts of personal data are utilized by neural models, promoting the protection of user privacy is of significant importance. In this work, we focus on unlearning in pre-trained LLMs. Our goal is to enable LLMs to unlearn particular training sequences while preserving the model's utility. The evaluation datasets are compiled from publicly accessible sources, adhering to the licenses associated with the collected data. We hope this work serves as a foundation for developing more practical MU methods for LLMs.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 62272183, 62071192 and 62171189; by the National Key R&D Program of China under Grant 2024YFE0103800; by the Major Science and Technology Project of Hubei Province under Grants 2024BAA008 and 2024BAA011; and by the Key R&D Program of Hubei Province under Grants 2024BAB016, 2024BAB031 and 2023BAB074.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of ACM CCS*, pages 308–318.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *CoRR, arXiv:1905.13319.*

George-Octavian Bărbulescu and Peter Triantafillou. 2024. To each (textual sequence) its own: Improving memorized-data unlearning in large language models. In *Proceedings of ICML.*

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of AAAI*, pages 7432–7439.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58(2).

Rob Bonta. 2022. California consumer privacy act (ccpa). *Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa.*

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *Proceedings of IEEE S&P*, pages 141–159.

Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *Proceedings of IEEE S&P*, pages 463–480.

Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. 2024. Learning to unlearn: Instance-wise unlearning for pretrained classifiers. In *Proceedings of AAAI*, pages 11186–11194.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2024a. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. In *Proceedings of NAACL*, pages 3190–3211.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024b. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Juan Manuel Zambrano Chaves, Eric Wang, Tao Tu, Eeshit Dhaval Vaishnav, Byron Lee, S Sara Mahdavi, Christopher Semturs, David Fleet, Vivek Natarajan, and Shekoofeh Azizi. 2024. Tx-llm: A large language model for therapeutics. *CoRR, arXiv:2406.06316.*

Jiaao Chen and Diyi Yang. 2023. Unlearn what you want to forget: Efficient unlearning for llms. In *Proceedings of EMNLP*, pages 12041–12052.

Ruizhe Chen, Tianxiang Hu, Yang Feng, and Zuozhu Liu. 2024. Learnable privacy neurons localization in language models. *CoRR, arXiv:2405.10989.*

Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2023. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of AAAI*, pages 7210–7217.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *CoRR, arXiv:1803.05457.*

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of ACL*, pages 8493–8502.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *CoRR, arXiv: 1811.01241.*

Yijiang River Dong, Hongzhou Lin, Mikhail Belkin, Ramon Huerta, and Ivan Vulić. 2024. Unmemorization in large language models via self-distillation and deliberate imagination. *CoRR, arXiv:2402.10052.*

Ronen Eldan and Mark Russinovich. 2023. Who's harry potter? approximate unlearning in llms. *CoRR, arXiv:2310.02238.*

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/toymodel/index.html.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Dennis Wei, Eric Wong, and Sijia Liu. 2024. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *Proceedings of ICLR*.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of AAAI*, pages 12043–12051.

Chongyang Gao, Lixu Wang, Chenkai Weng, Xiao Wang, and Qi Zhu. 2024. Practical unlearning for large language models. *CoRR, arXiv:2407.10223.*

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others.

2020. The pile: An 800gb dataset of diverse text for language modeling. *CoRR, arXiv:2101.00027.*

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of EMNLP*, pages 30–45.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of EMNLP*, pages 5484–5495.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. 2021. Mixed-privacy forgetting in deep networks. In *Proceedings of IEEE/CVF CVPR*, pages 792–801.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312.

A. Gordon, Z. Kozareva, and M. Roemmele. 2012. Semeval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In * *SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2020. Certified data removal from machine learning models. In *Proceedings of ICML*, pages 3832–3842.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in gpt2 language models. *CoRR, arXiv:2401.12181.*

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *CoRR, arXiv:2305.01610.*

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of IEEE/CVF CVPR*, pages 770–778.

Mark He Huang, Lin Geng Foo, and Jun Liu. 2024. Learning to unlearn for robust machine unlearning. *CoRR, arXiv:2407.10494.*

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *Proceedings of ICLR.*

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of ACL*, pages 14389–14408.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. *CoRR, arXiv:1909.06146.*

Aly Kassem, Omar Mahmoud, and Sherif Saad. 2023. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *Proceedings of the EMNLP*, pages 4360–4379.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, volume 1, page 2.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2021. Internet-augmented dialogue generation. *CoRR, arXiv: 2107.07566.*

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. 2024. Towards unbounded machine unlearning. In *Proceedings of NeurIPS*, volume 36.

Dohyun Lee, Daniel Rim, Minseok Choi, and Jaegul Choo. 2024. Protecting privacy through approximating optimal parameters for sequence unlearning in language models. *CoRR, arXiv: 2406.14091.*

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R Varshney, and 1 others. 2024a. Rethinking machine unlearning for large language models. *CoRR, arXiv:2402.08787.*

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, and 1 others. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017.

Zhe Liu and Ozlem Kalinli. 2024. Forgetting private textual sequences in language models via leave-one-out ensemble. In *Proceedings of ICASSP*, pages 10261–10265.

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

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *Proceedings of ICLR.*

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2022. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of ICLR.*

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

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023a. Membership inference attacks against language models via neighbourhood comparison. In *Proceedings of ACL Findings*, pages 11330–11343.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023b. Membership inference attacks against language models via neighbourhood comparison. In *Proceedings of ACL Findings*, pages 11330–11343.

Anmol Mekala, Vineeth Dorna, Shreya Dubey, Abhishek Lalwani, David Koleczek, Mukund Rungta, Sadid Hasan, and Elita Lobo. 2024. Alternate preference optimization for unlearning factual knowledge in large language models. *CoRR, arXiv:2409.13474*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Proceedings of NeurIPS*, pages 17359–17372.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2022. Pointer sentinel mixture models. In *Proceedings of ICLR*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *CoRR, arXiv:2402.06196*.

Jun Niu, Peng Liu, Xiaoyan Zhu, Kuo Shen, Yuecong Wang, Haotian Chi, Yulong Shen, Xiaohong Jiang, Jianfeng Ma, and Yuqing Zhang. 2024. A survey on membership inference attacks and defenses in machine learning. *Journal of Information and Intelligence*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *CoRR, arXiv:1606.06031*.

Mohaimenul Azam Khan Raiaan, Md Saddam Hossain Mukta, Kaniz Fatema, Nur Mohammad Fahad, Sadman Sakib, Most Marufatul Jannat Mim, Jubaer Ahmad, Mohammed Eunus Ali, and Sami Azam. 2024. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *CoRR, arXiv:1811.00207*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember what you want to forget: Algorithms for machine unlearning. *Proceedings of NeurIPS*, 34:18075–18086.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. *CoRR, arXiv: 2004.08449*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *CoRR, arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *CoRR, arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555.

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 ACL*, pages 13264–13276.

Ga Wu, Masoud Hashemi, and Christopher Srinivasa. 2022. Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of AAAI*, pages 8675–8682.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. Depn: Detecting and editing privacy neurons in pretrained language models. In *Proceedings of EMNLP*.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. *CoRR, arXiv:2402.15159*.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2023. Large language model unlearning. In *Proceedings of NeurIPS Workshop SoLaR*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *CoRR, arXiv: 1905.07830*.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024a. Negative preference optimization: From catastrophic collapse to effective unlearning. *CoRR, arXiv:2404.05868*.

Zhaohan Zhang, Ziquan Liu, and Ioannis Patras. 2024b. Get confused cautiously: Textual sequence memorization erasure with selective entropy maximization. *CoRR, arXiv:2408.04983*.

Rui Zheng, Bao Rong, Yuhao Zhou, Di Liang, Sirui Wang, Wei Wu, Tao Gui, Qi Zhang, and Xuan-Jing Huang. 2022. Robust lottery tickets for pre-trained language models. In *Proceedings of ACL*, pages 2211–2224.

A Related Work

A.1 Machine Unlearning

MU is first introduced by Cao et al. (Cao and Yang, 2015) and has since been widely studied in image classification tasks. MU aims to modify model parameters to eliminate the influence of specific data points or classes in image classification without requiring the model to be retrained from scratch after removing the unlearned data. In the literature, existing MU methods fall into two categories: exact unlearning and approximate unlearning.

Exact unlearning involves retraining the model from scratch after removing specific training data points. Bourtoule et al. (Bourtoule et al., 2021) propose SISA, which divides training data into shards, trains sub-models on each shard, and aggregates them. When an unlearning request arrives, only the relevant fragment is retrained. However, this approach comes with significant computational demands and requires access to the entire training set (Liu et al., 2024a). Wu et al. (Wu et al., 2022) introduce a framework that models how each individual training sample affects the model's performance across various criteria, and it removes the influence of the unlearned data. Golatkar et al. (Golatkar et al., 2021) propose an unlearning method, where the model is split into two parts: one that retains core data that does not need to be unlearned and remains unchanged, and another that undergoes unlearning with bounded parameters. While these methods are computationally efficient, they require extra storage to keep the intermediate parameters of different model slices and their corresponding training subsets.

To address these challenges, recent works have shifted towards scalable and effective approximate unlearning methods. Guo et al. (Guo et al., 2020) and Sekhari et al. (Sekhari et al., 2021) propose provably efficient unlearning methods based on Newton update removal mechanisms, leveraging the concept of differential privacy (DP) (Abadi et al., 2016). Golatkar et al. (Golatkar et al., 2020) propose Fisher Forgetting to scrub the information about the unlearned data by adding noise, calculated based on influence functions, to the parameters of the DNN model. However, these algorithms require the computation of second-order information (Hessian) of the loss function, which is prohibitively expensive due to the size of the parameters. Chundawat et al. (Chundawat et al., 2023) achieve unlearning by minimizing the divergence between a randomly initialized teacher model and the student model on unlearning data. Kurmanji et al. (Kurmanji et al., 2024) introduce a scalable unlearning model by using a teacher-student framework to selectively forget data, while addressing scalability and performance issues in MU. Huang et al. (Huang et al., 2024) adopt a meta-learning approach to optimize the unlearning process by leveraging feedback from a small subset of the remaining data. Yet, these training-based methods can get inefficient when the original dataset gets larger in scale. In addition, some research shifts the spotlight from the entire model parameters to specific, influential weights. Foster et al. (Foster et al., 2024) propose Selective Synaptic Dampening (SSD) method that identifies specific model parameters crucial for memorizing certain data by calculating the fisher information matrix. Fan et al. (Fan et al., 2024) propose saliency unlearning (SalUn) that identifies specific model weights by calculating the weight saliency map.

A.2 Machine Unlearning in LLMs

Given the vast amount of data and the large scale of the models involved in the training process, traditional MU methods are often impractical for application to LLMs due to their computational demands. LLM unlearning aims to remove the memory of training examples that users request to be erased from LLMs after training is completed, without affecting the model's performance on other data. Jang et al. (Jang et al., 2023) simply change the gradient descent to the opposite direction during language modeling to increase the model errors on the textual sequences to be unlearned. Buarbulescu et al. (Bărbulescu and Triantafillou, 2024) propose SGA, which performs selective gradient ascent during the unlearning process based on the degree of memorization of each unlearned text sequence within the LLM, continuing until there are no more elements above the unacceptable memorization threshold. Unfortunately, Maini et al. (Maini et al., 2024) demonstrated that gradient ascent itself in these methods fails to provide a satisfactory balance between forget quality and model utility.

To address this issue, some works assume that retained data can be utilized to maintain model utility (Wang et al., 2023; Chen and Yang, 2023; Yao et al., 2024; Zhang et al., 2024a). For example, Chen et al. (Chen and Yang, 2023) simultaneously perform gradient ascent on the unlearned data while minimizing the KL-divergence between the output from the updated model and the original model on the retained data, in order to maintain model performance in downstream classification and generation tasks. Similarly, Yao et al. (Yao et al., 2024) execute gradient descent or compute KL-divergence on the retained data to balance unlearning effectiveness with utility.

However, obtaining the retained dataset is often infeasible (Bărbulescu and Triantafillou, 2024; Yao et al., 2024). Additionally, Yao et al. (Yao et al., 2023) and Liu et al. (Liu et al., 2024b) propose using normal datasets (e.g., TruthfulQA) to ensure that unlearning harmful knowledge does not jeopardize responses to non-harmful prompts. However, there is no clear consensus on how to select a normal dataset that ensures the model's utility remains unaffected.

It can be observed that most LLM unlearning methods require fine-tuning the model to update parameters and need additional retained data to maintain model performance. However, in real-world applications, fine-tuning LLMs typically demands substantial resources for costly computations, and the retained data is often difficult to obtain due to privacy and storage issues. Against this background, we consider a highly constrained scenario and propose a practical unlearning method that only requires access to the unlearned data without necessitating fine-tuning the whole model.

B Experiment Setup

B.1 Models and Datasets.

Pre-trained LLMs. In our experiments, we follow existing MU works (Jang et al., 2023; Kassem et al., 2023; Bărbulescu and Triantafillou, 2024) and use the GPT-Neo model family (125M, 1.3B, 2.7B) (Black et al., 2021), pre-trained on the Pile dataset (Gao et al., 2020), to perform a fair comparison with existing works.

Unlearned Data. The unlearned samples are selected randomly from the Training Data Extraction Challenge[3] (Jang et al., 2023). This challenge provides a dataset containing 15,000 samples from the pre-trained GPT-Neo family models. Each sample consists of 200 tokens and has been shown to be memorized by the GPT-Neo models, making them hard to be unlearned.

B.2 Evaluation Metrics.

We evaluate NeuMuter in terms of both unlearning effectiveness and model utility.

From the effectiveness perspective, we use memorization accuracy (Jang et al., 2023) and extraction likelihood (Jang et al., 2023) to quantify the memorization of LLMs with respect to the given unlearned data. Additionally, we perform a loss-based membership inference attack (MIA) on the unlearned LLMs to assess how much information about the unlearned data remains in the model, following the approach in (Bărbulescu and Triantafillou, 2024).

Memorization Accuracy (MA). This metric measures the ratio at which the model's prediction of the next token matches the original token in the unlearned data:

$$\mathrm{MA}(\mathbf{x}) = \frac{\sum_{t=1}^{T-1} \mathbb{1}[\mathrm{argmax}(p_\theta(\cdot|w_{<t}) = w_t]}{T-1}. \tag{15}$$

Extraction Likelihood (EL). This metric measures the overlap between the generated output of the unlearned LLM $\mathcal{M}_\theta(w_{<t})$ and the unlearned sample $w_{\geq t}$:

$$\mathrm{EL}_n(\mathbf{x}) = \frac{\sum_{t=1}^{T-n} \mathrm{Overlap}_n(\mathcal{M}(w_{<t}), w_{\geq t})}{T-n}, \tag{16}$$

where $\mathrm{Overlap}_n$ measures the length of the n-gram overlap between the unlearned sample and the generated output.

As for the utility, we quantify the performance of the unlearned LLMs using various metrics, including classification accuracy on NLP classification tasks, F1 score and cross-entropy loss on dialogue

[3] https://github.com/google-research/lm-extraction-benchmark

reasoning benchmarks, and perplexity on the validation corpus.

Classification Accuracy (ACC). We use standard classification accuracy in ML field. Specifically, we measure accuracy on 9 popular NLP classification tasks, including ARC-Easy and ARC-Challenge (Clark et al., 2018), COPA (Gordon et al., 2012), Lambada (Paperno et al., 2016), Winogrande (Sakaguchi et al., 2021), PubMedQA (Jin et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and MathQA (Amini et al., 2019). Consistent with established practices, we report the average accuracy across all classification tasks.

F1 Score (F1) and *Cross-Entropy (CE).* We evaluate the generation capabilities of the unlearned LLMs on four dialogue benchmarks: Wizard of Wikipedia (WoW) (Dinan et al., 2018), Wizard of Internet (WoI) (Komeili et al., 2021), Blended Skill Talk (BST) (Smith et al., 2020), and Empathetic Dialogues (ED) (Rashkin et al., 2018). For each dialogue, we use the standard F1 Score and calculate the CE loss of the generated outputs from the unlearned LLMs with respect to the ground truth content.

Perplexity (PPL). To assess the quality of the generated outputs from the unlearned LLMs, we calculate the perplexity on the validation corpora of Pile (Gao et al., 2020) and Wikitext (Merity et al., 2022) to evaluate the utility of the unlearned models.

B.3 MIA Settings

MIAs aim to predict whether a given data sample was used to train the target model. Therefore, the performance of MIAs can provide insight into how much information about the unlearned data is retained by the unlearned LLM. In our experiment, we conduct a loss-based MIA for unlearning in LLMs, following the latest work (Bărbulescu and Triantafillou, 2024). The intuition is that if the unlearned data did not contribute to the model's training, the loss of this data should be similar to the loss of the validation data, which was not part of the original training dataset. Therefore, we first apply the unlearning methods to the original model to forget 32 samples, and then conduct the MIA to the unlearned models. We perform the above process on models of different sizes and compare the MIA results of the unlearned models with the original model. We calculate the distance between the loss on the unlearned data and the average loss on the validation data, which is generated using a bidirectional encoder in (Bărbulescu and Triantafillou, 2024). A smaller distance indicates that the MIA will be less successful, suggesting better privacy protection for the unlearned LLM. Conversely, a larger distance implies that the MIA will be more successful, leading to more privacy leakage.

B.4 Implementation Details.

Implementation Details. Following existing work (Jang et al., 2023), we define an unlearned sample $\mathbf{x}$ as having been forgotten if its $\mathrm{EL}_n(\mathbf{x})$ and $\mathrm{MA}(\mathbf{x})$ values are lower than the average EL_n and MA values for samples that were not seen during training. We continue applying both the comparison methods and NeuMuter until all unlearned samples meet the threshold (i.e., all methods achieve consistent unlearning performance on the unlearned data.), defined as:

$$\begin{aligned}\mathrm{EL}_n(\mathbf{x}) &\leq \frac{1}{|D'|}\sum_{\mathbf{x}'\in D'}\mathrm{EL}_n(\mathbf{x}'),\\ \mathrm{MA}(\mathbf{x}) &\leq \frac{1}{|D'|}\sum_{\mathbf{x}'\in D'}\mathrm{MA}(\mathbf{x}'),\end{aligned} \tag{17}$$

where D' represents a validation corpus not seen during training. We obtain the threshold (Jang et al., 2023), which is the average EL_{10} and MA (Eq. 17), by calculating from 10,000 randomly selected instances from the Pile validation corpus, and the results are shown in Table 11.

As for the settings for the comparisons, we use the same learning rate of 5e-4 for GA as in its original paper (Jang et al., 2023), since unlearning is performed over the same data distribution. For SGA, we set the memorization upper limit to $y = 70\%$. If the average memorization across the population of textual sequences does not reach this level, we continue unlearning by applying gradient ascent to the top-k (where $k = 1$) most memorized samples from $\mathcal{D}_f$, as described in the original paper (Bărbulescu and Triantafillou, 2024). For methods that require the setting of unlearning strength τ, such as TA and DI, we set τ to 3 and 8, respectively, to achieve average EL_{10} and MA values below the unlearning threshold.

For NeuMuter, we train the learnable mask for $E = 100$ iterations, using hyperparameters $\lambda = 500$ and $\eta = 5$. We treat the hyperparameters β, γ, and ξ as constants, following (Louizos et al., 2018), with values $\beta = 0.3$, $\gamma = -0.1$, and $\xi = 1.1$. We generate $K = 5$ neighbor samples for each unlearned text using a pre-trained BERT (Kenton and Toutanova, 2019), with a replacement ratio of $r = 1$.

In all experiments, we set the number of unlearned samples to 32 by default and randomly select them from the dataset mentioned above. We also perform five trials for each experiment and present the average results across all experimental settings. Lastly, in terms of hardware, all experiments are conducted on two NVIDIA GeForce RTX 4090 GPUs, each with 24 GB of memory.

C Additional Experimental Results

C.1 Performance on Downstream Tasks

We provide detailed experimental results for Table 2 in the main text, including the accuracy on 9 text classification tasks as shown in Table 4, the F1 Score and CE on 4 text dialogue tasks as shown in Table 5 and Table 6, and the PPL on 2 validation corpora as shown in Table 7.

In Table 4, we can see that NeuMuter consistently outperforms all other methods across multiple classification tasks and model sizes. For models with parameters of 125M, 1.3B, and 2.7B, it achieves the highest average accuracy of 42.67%, 49.87%, and 52.48%, respectively, surpassing the other methods like GA, IG, and DI. Notably, NeuMuter excels in reasoning tasks (e.g., ARC-E, ARC-C) and commonsense reasoning tasks (e.g., Piqa, MathQ), with significant improvements in larger models. Its robustness and consistent performance across tasks make it a strong unlearning approach for language models, particularly when scaling to larger sizes.

In Table 5 and Table 6, the F1 scores and CE loss for dialogue tasks demonstrate that NeuMuter consistently outperforms or is competitive with other methods across all models. In terms of F1 scores, NeuMuter achieves the highest average performance, with scores of 10.23, 12.10, and 12.39 for the GPT-Neo models with 125M, 1.3B, and 2.7B parameters, respectively. These scores are notably higher than those of GA, SGA, and IG, and are comparable to or slightly better than DI, indicating NeuMuter's superior ability to preserve the model utility in generating balanced predictions across a variety of dialogue tasks.

Furthermore, NeuMuter also excels in minimizing CE loss, with values of 3.83, 3.26, and 3.14 for the same models. These results are lower than those of GA, SGA, and IG, further illustrating NeuMuter's capacity to reduce prediction error and improve model efficiency. Overall, NeuMuter's strong performance in both F1 score and CE loss highlights its effectiveness as a robust unlearning method.

Table 7 demonstrates the effect of various unlearning methods, including NeuMuter, on the perplexity of models evaluated on the Pile and WikiText corpora. NeuMuter consistently shows strong performance, achieving lower average PPL compared to most other methods. For the GPT-Neo model with 125M parameters, NeuMuter achieves an average PPL of 39.17, significantly outperforming methods like GA (638.23), SGA (803.82), and IG (200.85), and also surpassing TA (120.15) and DI (39.90). For the 1.3B model, NeuMuter leads with a PPL of 17.92, outshining GA (7103222.30) and IG (246.38), and performing similarly to TA (47.20) and SGA (50.91). In the case of the 2.7B model, NeuMuter achieves an average PPL of 16.52, slightly better than DI (16.88) and much lower than GA (5231639.15) and IG (341.15), with TA (24.15) and SGA (298.52) performing worse. Overall, NeuMuter outperforms GA and IG and competes well with other methods like DI and TA, suggesting it effectively reduces perplexity while maintaining model efficiency.

C.2 Generation Examples of Unlearned LLMs

We record the output of the LLMs before and after applying NeuMuter and compare the differences in the generated content with respect to the same unlearned sample. Specifically, we take the first 100 tokens of each unlearned sample (with a total length of 200 tokens) as the prefix and ask both the original and unlearned LLMs to generate the next 100 tokens as the suffix. We then compare the overlap between the models' outputs, both before and after unlearning, with the correct suffix.

Table 14 presents three examples to demonstrate how NeuMuter protects against extraction attacks. The results show that the original model generates content that perfectly matches the correct suffix. After applying NeuMuter unlearning, the model generates content that is contextually similar to the original suffix based on the prefix but distinct from the correct suffix.

C.3 Efficiency of NeuMuter

Efficiency is a crucial factor in evaluating the practicality of MU methods. To demonstrate the ef-

Model	Method	Hella.	Lamba.	Wino.	COPA	ARC-E	ARC-C	Piqa	MathQ	PubQ	Avg.
GPT-Neo (125M)	Original	28.19	37.60	51.52	59.80	45.96	22.41	63.26	21.51	57.60	43.09
	GA	27.34	35.56	51.95	60.20	37.89	20.40	60.20	21.77	57.60	41.43
	TA	26.70	1.46	51.44	52.60	36.67	20.40	57.50	20.8	57.00	36.06
	SGA	27.38	32.72	51.27	60.20	37.02	19.40	60.27	22.54	57.60	40.93
	DI	28.2	36.69	51.67	59.39	44.52	22.86	62.59	20.72	57.39	42.67
	IG	27.63	35.84	52.26	56.60	36.14	18.06	58.25	22.11	57.60	40.50
	NeuMuter	28.20	36.02	51.87	60.60	45.09	23.08	62.81	21.24	57.60	42.67
GPT-Neo (1.3B)	Original	36.98	57.34	54.86	72.20	56.49	25.75	70.22	22.47	53.80	50.01
	GA	35.16	54.92	54.77	68.20	54.21	24.75	68.31	22.34	52.80	48.38
	TA	31.06	41.52	52.46	62.00	44.04	23.41	63.86	21.94	57.60	44.21
	SGA	33.50	52.96	52.29	67.50	53.68	23.07	64.11	21.51	54.20	46.98
	DI	35.60	51.12	51.96	69.60	54.91	26.09	68.62	22.68	54.8	48.37
	IG	27.42	5.68	51.20	55.80	35.79	18.73	57.60	22.91	54.20	36.59
	NeuMuter	36.97	58.22	53.84	71.20	55.79	26.48	69.40	22.31	54.60	49.87
GPT-Neo (2.7B)	Original	40.78	62.28	56.63	74.80	59.47	25.75	72.88	23.61	57.00	52.58
	GA	34.40	61.58	54.70	72.00	53.51	26.42	69.44	22.28	57.60	50.21
	TA	35.31	58.86	54.26	72.40	54.74	22.74	67.92	22.78	57.20	49.58
	SGA	36.50	61.68	55.50	73.20	57.09	26.80	70.88	23.39	57.60	51.40
	DI	40.04	60.28	55.56	74.20	58.95	25.75	73.02	23.01	56.40	51.91
	IG	28.87	31.33	51.22	57.20	36.14	21.40	58.38	21.64	32.40	37.62
	NeuMuter	40.77	61.46	55.72	73.80	60.00	27.09	72.51	23.58	57.40	52.48

Table 4: Classification accuracy on 9 text classification tasks.

Model	Method	WoW	ED	BST	WoI	Avg.
GPT-Neo (125M)	Original	10.49	8.38	9.66	11.24	9.94
	GA	0.68	0.54	0.79	0.67	1.42
	TA	5.72	5.28	5.54	6.04	5.64
	SGA	0.59	0.45	0.69	0.55	1.32
	DI	10.66	8.13	9.01	11.5	9.83
	IG	5.70	6.00	4.50	6.00	5.55
	NeuMuter	10.97	8.79	9.94	11.22	10.23
GPT-Neo (1.3B)	Original	12.70	10.50	12.13	13.70	12.26
	GA	12.16	9.43	11.21	13.08	11.47
	TA	7.66	5.65	6.43	8.71	7.11
	SGA	12.56	9.76	11.62	13.45	11.85
	DI	10.56	7.45	8.78	11.86	9.66
	IG	10.12	9.01	8.38	9.43	9.24
	NeuMuter	12.63	10.26	11.91	13.60	12.10
GPT-Neo (2.7B)	Original	12.45	10.82	12.45	13.61	12.33
	GA	12.52	10.02	10.91	12.36	11.45
	TA	10.04	6.93	8.11	10.53	8.90
	SGA	12.51	10.23	11.11	12.40	11.56
	DI	12.31	10.46	11.82	13.60	12.05
	IG	9.26	7.59	6.04	8.48	7.84
	NeuMuter	12.51	10.98	12.35	13.71	12.39

Table 5: F1 scores on 4 text dialogue tasks

Model	Method	WoW	ED	BST	WoI	Avg.
GPT-Neo (125M)	Original	3.82	3.72	3.81	3.73	3.77
	GA	6.93	6.60	6.75	6.80	6.77
	TA	4.94	4.8	4.79	4.73	4.81
	SGA	7.12	6.80	6.93	6.99	6.96
	DI	3.84	3.73	3.80	3.75	3.78
	IG	5.25	5.00	5.10	5.04	5.10
	NeuMuter	3.87	3.79	3.88	3.78	3.83
GPT-Neo (1.3B)	Original	3.27	3.23	3.29	3.22	3.25
	GA	3.37	3.40	3.37	3.30	3.36
	TA	4.17	4.36	4.19	4.04	4.19
	SGA	3.35	3.39	3.36	3.28	3.30
	DI	3.34	3.28	3.29	3.28	3.30
	IG	5.70	5.64	5.77	5.60	5.68
	NeuMuter	3.29	3.24	3.30	3.22	3.26
GPT-Neo (2.7B)	Original	3.26	3.28	3.30	3.17	3.25
	GA	3.78	3.97	3.77	3.59	3.78
	TA	3.73	3.82	3.79	3.62	3.74
	SGA	3.55	3.64	3.54	3.46	3.55
	DI	3.26	3.26	3.28	3.17	3.25
	IG	6.04	6.06	6.03	5.90	6.01
	NeuMuter	3.16	3.14	3.17	3.09	3.14

Table 6: CE loss on 4 text dialogue tasks.

ficiency of NeuMuter, we measure the time consumption required for NeuMuter and the baselines to unlearn 32 samples using the GPT-Neo models. We also analyze the proportion of the GPT-Neo models' parameters that need to be modified by each method, with the results summarized in Table 9.

From these results, we can see that NeuMuter requires the minimal computational overhead compared with existing MU methods. For the 1.3B parameter model, NeuMuter achieves the lowest MA of 30.89% while requiring only 9.68 minutes. Since NeuMuter only inserts trainable masks into the FFN module of each Transformer block, the number of trainable parameters in NeuMuter is proportional to the number of layers and the dimensions of the FFN's hidden states. For the GPT-Neo models with 125M, 1.3B, and 2.7B parameters, the

Model	Method	WoW	ED	BST	WoI	Avg.
GPT-Neo (125M)	Original	3.82	3.72	3.81	3.73	3.77
	GA	6.93	6.60	6.75	6.80	6.77
	TA	4.94	4.8	4.79	4.73	4.81
	SGA	7.12	6.80	6.93	6.99	6.96
	DI	3.84	3.73	3.80	3.75	3.78
	IG	5.25	5.00	5.10	5.04	5.10
	NeuMuter	3.87	3.79	3.88	3.78	3.83
GPT-Neo (1.3B)	Original	3.27	3.23	3.29	3.22	3.25
	GA	3.37	3.40	3.37	3.30	3.36
	TA	4.17	4.36	4.19	4.04	4.19
	SGA	3.35	3.39	3.36	3.28	3.30
	DI	3.34	3.28	3.29	3.28	3.30
	IG	5.70	5.64	5.77	5.60	5.68
	NeuMuter	3.29	3.24	3.30	3.22	3.26
GPT-Neo (2.7B)	Original	3.26	3.28	3.30	3.17	3.25
	GA	3.78	3.97	3.77	3.59	3.78
	TA	3.73	3.82	3.79	3.62	3.74
	SGA	3.55	3.64	3.54	3.46	3.55
	DI	3.26	3.26	3.28	3.17	3.25
	IG	6.04	6.06	6.03	5.90	6.01
	NeuMuter	3.16	3.14	3.17	3.09	3.14

Table 7: PPL on 2 validation corpora.

K	MA (%) ↓	Acc (%) ↑	F1 (%) ↑	CE ↓	PPL ↓
1	30.89	**49.87**	12.10	3.26	17.92
2	32.47	49.09	12.18	3.26	**17.61**
3	30.99	49.65	12.15	3.27	17.75
4	31.45	49.77	12.13	3.28	17.82
5	**30.20**	49.86	**12.19**	**3.26**	17.86

Table 8: Impact of the number of neighbor samples K on NeuMuter with respect to GPT-Neo (1.3B).

Model	Method	MA (%) ↓	Time Consumed (Minutes)↓	Trainable Params. (%)↓
GPT-Neo (125M)	GA	29.43	6.12	100
	TA	29.73	7.33	100
	SGA	29.37	8.42	100
	DI	29.33	6.30	100
	IG	29.11	160.96	0.0
	NeuMuter	**28.63**	**2.40**	0.03
GPT-Neo (1.3B)	GA	31.01	12.50	100
	TA	31.74	12.45	100
	SGA	30.98	16.80	100
	DI	31.11	10.20	100
	IG	31.97	1162.24	0.0
	NeuMuter	**30.89**	**7.68**	0.01
GPT-Neo (2.7B)	GA	33.04	16.20	100
	TA	33.10	16.05	100
	SGA	32.85	18.10	100
	DI	33.33	16.00	100
	IG	33.83	2080.80	0.0
	NeuMuter	**32.79**	**12.16**	0.01

Table 9: Time consumption and trainable parameter ratio of NeuMuter and the comparisons.

r	MA (%) ↓	Acc (%) ↑	F1 (%) ↑	CE ↓	PPL ↓
0.01	31.64	48.67	12.04	3.27	18.79
0.1	33.12	49.06	12.04	3.27	18.28
0.2	**31.52**	48.59	12.05	3.28	18.60
0.4	33.23	49.44	12.13	3.26	18.35
0.6	32.03	49.35	12.10	3.26	18.25
0.8	33.23	49.20	12.08	**3.25**	18.00
1	32.44	**49.60**	**12.13**	3.26	**17.81**

Table 10: Impact of the number of replacement ratio r on NeuMuter with respect to GPT-Neo (1.3B).

trainable parameters of NeuMuter account for just 0.03%, 0.01%, and 0.01% of the total model parameters, respectively. In contrast, fine-tuning-based MU methods, such as GA, TA, SGA, and DI, require fine-tuning all model parameters. This is the main reason for that NeuMuter is able to efficiently achieve the goal of MU in LLMs.

In addition, we observe that SGA incurs higher computational overhead than GA under the same conditions due to the need to test the memorization level of each sample at the end of each epoch. IG exhibits the worst performance in terms of computational complexity. Although IG does not require model fine-tuning or the insertion of additional trainable parameters, it requires multiple gradient computations for each token in every sample, resulting in substantial time costs. This makes IG almost impractical for unlearning tasks in LLMs.

C.4 Impact of Neighbor Sample Generation

In this section, we evaluate the impact of the neighbor samples on the performance of our NeuMuter. We conduct experiments focusing on two main hyper-parameters of NeuMuter: the number of neighbor samples K used to maintain model performance and the replacement ratio r of original tokens in the unlearned samples.

C.4.1 Impact of Size of Neighbor Samples

We apply NeuMuter to GPT-Neo with 1.3B parameters to unlearn 32 samples, while keeping other hyperparameters fixed and varying K for each target sample from 1 to 5. We report the model's MA on the forgotten set after each unlearning process, as well as the average accuracy across 9 text classification tasks, and the average F1 score and CE loss across 4 dialogue tasks. As shown in Table 8,

Model (Size)	EL_{10} (%) Threshold	MA (%) Threshold
GPT-Neo (125M)	4.99	29.94
GPT-Neo (1.3B)	5.68	33.27
GPT-Neo (2.7B)	5.53	34.02

Table 11: Threshold of EL_{10} and MA Metrics for GPT-Neos.

Model	Neighbors	Validation Corpus
GPT-Neo (125M)	29.34	29.94
GPT-Neo (1.3B)	33.11	33.27
GPT-Neo (2.7B)	33.66	34.02

Table 12: MA on Neighbor Samples vs. Validation Corpus.

increasing K slightly improves NeuMuter's performance on the remaining data. When K is set to 5, the unlearned model shows improvements in F1 score and PPL after applying NeuMuter, but the gains are limited. Specifically, increasing K leads to higher computational time costs. When computational resources are limited, using fewer neighbors helps strike a better balance between performance and efficiency.

C.4.2 Impact of Replacement Radio of Neighbor Samples

Similarly, we apply NeuMuter to GPT-Neo (1.3B) model to unlearn 32 samples, while varying r of original tokens in the generation of neighbors from 0.01 to 1, keeping other hyperparameters fixed. From the results in Table 10, we can see that when r is very low, NeuMuter significantly impacts the model's performance after forgetting the target samples. This occurs because the generated neighbors retain much of the knowledge from the original text, leading to instability during the mask training process. As a result, the utility of the unlearned model is relatively reduced while achieving comparable forgetting performance. As r increases, more key information from the original text is replaced, which helps obscure the specific characteristics of the original data. For example, when $r = 1$, the unlearned model's accuracy on text classification tasks improves by 1% compared to when $r = 0.01$. This is because as the replacement ratio increases, more key information from the original text is replaced, allowing NeuMuter to better identify neurons that are highly relevant to the unlearned data, which in turn enhances the stability of the mask training.

C.4.3 Information Leakage of Neighbor Samples

Actually, the neighborhood dataset we created does not cause any information leakage from the unlearned data. We use a Masked Language Model (MLM, e.g., BERT), whose training data excludes the unlearned data, to generate a neighborhood sample for each unlearned sentence. Specifically, we start from the first token of the unlearned sentence and proceed sequentially to the last, masking one token at a time and using BERT to predict its replacement. Since BERT is not trained on the unlearned data and thus is unaware of the unlearned data's information, the generated neighbors can replace sensitive content with benign alternatives while preserving grammatical structure and general semantic context.

To further verify that the neighborhood dataset does not involve information leakage, we evaluate the memorization accuracy (MA) of the target model on the neighborhood dataset as shown in table 12. The MA values on the neighborhood dataset are comparable to or lower than those on the validation corpus, indicating that no information leakage has occurred.

D Additional Information on TOFU

Very recently, the TOFU benchmark (Maini et al., 2024) is proposed to investigate the issue of whether the unlearned model and the retrained model are indistinguishable, and is adopted for evaluation by many studies (Zhang et al., 2024a; Gao et al., 2024; Zhang et al., 2024b; Mekala et al., 2024). Therefore, we also use the TOFU benchmark to further evaluate the performance of our NeuMuter and compare it with the MU methods included in the benchmark.

D.1 Experiment Setup

Dataset. The TOFU benchmark (Maini et al., 2024) provides a dataset of synthetically generated biographies for 200 fictional authors, created by GPT-4. Each biography consists of 20 question-answer pairs. A subset of these profiles, called the “forget set", serves as the unlearned data, while the remaining profiles are referred to as the “retain set", which the model is expected to retain after unlearning. Specifically, the MU task involves unlearning

1%, 5%, and 10% of the biographies from an LLM that has already been fine-tuned on data from all 200 authors.

Additionally, TOFU includes two datasets related to real-world authors and world facts from various domains, which are used to test the retention of useful knowledge after LLM unlearning. The Real Authors set includes question-answer pairs about authors in the real world and often deals with neighbor concepts entangled with those in the unlearned set. The Real World set contains commonsense knowledge about the real world and is designed to test the performance on general world knowledge of the unlearned model.

Pre-trained LLMs. In the TOFU benchmark, since the synthetic biographies have not participated in the training process of the pre-trained LLMs, we fine-tune the pre-trained LLM on this dataset to enable it to memorize these biographies. In this experiment, we use a fine-tuned Phi-1.5B[4] on the TOFU dataset as the targeted LLM that needs unlearning the forget set.

Evaluation Metrics. To measure the effectiveness of NeuMuter, we use the evaluation metrics proposed in the TOFU benchmark (Maini et al., 2024), which focus on Forget Quality and Model Utility.

Forget Quality measures how closely the output of the unlearned model matches the output of a retrained model that was trained only on the retained data, in distribution. We calculate the truth ratio on the unlearned set for both the retrained and unlearned models, producing two different distributions. Specifically, we perform the Kolmogorov-Smirnov test and compute the p-value. A large p-value indicates that the two models are indistinguishable in terms of their truth ratio.

Model Utility is aggregated as the harmonic mean of 9 results: the probability, ROUGE, and truth ratio from each of the three datasets: the retain set, the real author set, and the world facts set. Specifically, the definition of each metric is provided below.

- *Probability.* Given an input sequence of TOFU $x = [q, a]$, where q represents the question, a is the answer, and $|a|$ denotes the number of tokens in the answer. On the retain or forget set, we calculate the normalized conditional probability $P(a \mid q)^{1/|a|}$. On the real authors and world facts subsets, the TOFU dataset provides multiple answer choices $\{a_1, \cdots, a_n\}$ for each question q, where only one answer is correct. Assuming that a_1 is the correct answer, we compute the probability as follows:

$$Probability = \frac{P(a_1 \mid q)^{1/|a_1|}}{\sum_{i=2}^{n} P(a_i \mid q)^{1/|a_i|}}. \quad (18)$$

- *ROUGE.* We compute the ROUGE-L recall score (Lin, 2004) between the ground truth and the answers generated by the unlearned LLM.

- *Truth Ratio.* Specifically, For a given question, the truth ratio is computed to approximately compare how likely its correct answer is to an incorrect answer, formulated as:

$$R_{\text{truth}} = \frac{\frac{1}{|\mathcal{A}_{\text{pert}}|} \sum_{\hat{a} \in \mathcal{A}_{\text{pert}}} P(\hat{a} \mid q)^{1/|\hat{a}|}}{P(\tilde{a} \mid q)^{1/|\tilde{a}|}}, \quad (19)$$

where $\mathcal{A}_{\text{pert}} = \{\hat{a}_1, \hat{a}_2, \cdots\}$ represents the set of perturbed answers and $\tilde{a}$ is the paraphrased answer with GPT-4. The truth ratio takes the mean of the conditional probabilities of perturbed answers and normalizes it by the conditional probability of the correct paraphrased answer.

Baseline Methods. In this paper, we compare the performance of our NeuMuter against all baselines included in the TOFU benchmark (Maini et al., 2024), such as GA (described in Section 4), Gradient Difference (GD), KL Minimization (KL), and Preference Optimization (PO).

- *Gradient Difference (GD).* In contrast to GA, this method not only aims to increase the loss on the forget set $\mathcal{D}_f$, but also performs gradient descent on the retain set $\mathcal{D}_r$, thereby reducing the decrease in model utility.

- *KL Minimization (KL).* This method maintains the model's performance on the remaining data by minimizing the KL divergence between the predictions on $\mathcal{D}_r$ of the original (fine-tuned on TOFU) and the unlearned model, while achieving unlearning by maximizing the conventional loss on $\mathcal{D}_f$.

- *Preference Optimization (PO).* This method aims to adjust the model to avoid revealing information about specific authors by modifying its responses. It computes the loss for

[4] https://huggingface.co/locuslab/tofu_ft_phi-1.5

Size	Method	Model Utility ↑	Forget Quality ↑
	Original	0.62	-20.74
	Retrain	0.61	0.00
1%	GD	0.61	-2.17
	PO	**0.62**	-1.84
	GA	0.61	-1.84
	KL	0.61	-2.00
	NeuMuter	0.61	**-0.04**
5%	GD	0.47	-3.02
	PO	0.51	-8.97
	GA	0.35	**-0.24**
	KL	0.43	-0.53
	NeuMuter	**0.61**	-0.41
10%	GD	0.38	-5.31
	PO	0.30	-11.09
	GA	0.28	-8.73
	KL	0.00	-9.31
	NeuMuter	**0.59**	**-1.09**

Table 13: Performance of NeuMuter on TOFU benchmark with respect to LLaMA-2-7B model.

a question paired with an alternative answer, such as "I do not know the answer," to prevent the model from outputting information about the relevant author.

D.2 Performance on Larger Model

To further evaluate NeuMuter, we conduct experiments on a larger language model, LLaMA-2-7B, using synthetic unlearned data at varying scales (1%, 5%, and 10%). Table 13 reports the unlearning performance across these different unlearning proportions. Consistent with prior results on smaller models, we observe that NeuMuter outperforms all baseline methods in terms of both model utility and forget quality. Specifically, NeuMuter achieves utility scores comparable to the original model while significantly improving forget quality, closely approaching the performance of the retrained model.

When the unlearned set is small (1%), most baselines exhibit reasonable utility but suffer from poor forget quality. NeuMuter, by contrast, achieves a near-optimal balance. As the size of the unlearned data increases to 5% and 10%, many baselines (e.g., GA, KL, PO) either severely degrade in utility or fail to unlearn effectively. NeuMuter stands out by maintaining high utility even under larger unlearning demands, demonstrating both robustness and scalability.

Algorithm 1 NeuMuter

```
Input: Unlearned Data $\mathcal{D}_f$, LLM $\mathcal{M}_o$ with
       original parameter $\theta_o$, neighbors number $K$, replacement ratio $r$, trainable
       parameter $\boldsymbol{\alpha}$, mask threshold $\epsilon$, max iteration number $E$.
Output: Unlearned model $\mathcal{M}_u$ with parameter
       $\theta_u$.
  Memorization Neuron Localization
  $\tilde{\mathcal{D}}_n$ = Neighbor Samples Generation$(\mathcal{D}_f, K, r)$

  Initialize $\mathbf{M}$ with parameter $\boldsymbol{\alpha}$
  $\theta \leftarrow$ Add $\mathbf{M}$ to $\theta_o$
  Let $\mathcal{N} = \varnothing$
  for $epoch = 1$ to $E$ do
      Calculate $\mathcal{L}_{MU}$ on $\mathcal{D}_f$ and $\tilde{\mathcal{D}}_n$
      Optimize $\boldsymbol{\alpha}$ with $\mathcal{L}_{MU}$
  end for
  for $m_i^l$ in $\mathbf{M}$ do
      if $m_i^l < \epsilon$ then
          $\mathcal{N} \leftarrow \mathcal{N} \cup \mathbf{v}_i^l$
      end if
  end for
  return $\mathcal{N}$
  Memorization Removal
  Initialize $\theta_u \leftarrow \theta$
  for $l$ in $L$ do
      for $m_i^l$ in $\mathbf{m}^l$ do
          $m_i^l = \mathbb{1}_{\{\mathbf{v}_i^l \in \mathcal{N}\}} \cdot m_i^l + \mathbb{1}_{\{\mathbf{v}_i^l \notin \mathcal{N}\}}$
      end for
      $\mathbf{a}^l = \mathbf{a}^l \odot \mathbf{m}^l$
  end for
  return $\mathcal{M}_u$
```

E Pseudocode

The Pseudocode of NeuMuter is shown in Alg. 1.

Status	Text
Original Text	We work hard every day to put quality content out there for our community. Techdirt is one of the few remaining truly independent media outlets. We do not have a giant corporation behind us, and we rely heavily on our community to support us, in an age when advertisers are increasingly uninterested in sponsoring small, independent sites — especially a site like ours that is unwilling to pull punches in its reporting and analysis. While other websites have resorted to paywalls, registration requirements, and increasingly annoying/intrusive advertising, we have always kept Techdirt open and available to anyone. But in order to continue doing so, we need your support.
Before Unlearning	We work hard every day to put quality content out there for our community. Techdirt is one of the few remaining truly independent media outlets. We do not have a giant corporation behind us, and we rely heavily on our community to support us, in an age when advertisers are increasingly uninterested in sponsoring small, independent sites — especially a site like ours that is unwilling to pull punches in its reporting and analysis. While other websites have resorted to paywalls, registration requirements, and increasingly annoying/intrusive advertising, we have always kept Techdirt open and available to anyone. But in order to continue doing so, we need your support.
After Unlearning	We work hard every day to put quality content out there for our community. Techdirt is one of the few remaining truly independent media outlets. We do not have a giant corporation behind us, and we rely heavily on our community to support us, in an age when advertisers are increasingly uninterested in sponsoring small, GamLeanThreadsburgAlrightCNET In the past few years, the tech world has seen a lot of changes. The internet has become a lot more accessible, and the world has become more connected. But, there's still a lot of people who are still stuck in the past. One of the biggest changes in the tech world is the rise of the cloud. The cloud is a lot more accessible than ever before.
Original Text	Program</refentrytitle></citerefentry></para></refsect1><refsect1 xml:id=‘Copyright"><title>Copyright </title><para>Copyright <trademark class=‘copyright"/>2003-2005 3Dlabs Inc. Ltd. Copyright <trademark class=‘copyright"/>2010-2014 Khronos Group. This material may be distributed subject to the terms and conditions set forth in the Open Publication License, v 1.0, 8 June 1999. <link xmlns:xlink=“http://www.w3.org/1999"
Before Unlearning	Program</refentrytitle></citerefentry></para></refsect1><refsect1 xml:id=“Copyright"><title>Copyright </title><para>Copyright <trademark class=“copyright"/>2003-2005 3Dlabs Inc. Ltd. Copyright <trademark class=“copyright"/>2010-2014 Khronos Group. This material may be distributed subject to the terms and conditions set forth in the Open Publication License, v 1.0, 8 June 1999. <link xmlns:xlink=“http://www.w3.org/1999"
After Unlearning	Program</refentrytitle></citerefentry></para></refsect1><refsect1 xml:id=“Copyright"><title>Copyright </title><para>Copyright <trademark class=“copyright"/>2003-2005 3Dlabs Inc. Ltd. ThisCLUDedjavalicense 3Dlabs Inc. Ltd. </para> </refsect1> </refentry>

Table 14: Examples performing extraction attacks on token sequences. The coloured part denotes the model generated text given the prefix of length 100 as input. Green indicates that the part is generated based on a prefix that is identical to the original text, and red indicates that it is not. For the extraction attack, we utilize a naïve greedy decoding strategy.