

Private Memorization Editing: Turning Memorization into a Defense to Strengthen Data Privacy in Large Language Models

Elena Sofia Ruzzetti¹, Giancarlo A. Xompero^{1,2},
Davide Venditti¹, Fabio Massimo Zanzotto^{1,2}

¹Human Centric ART, University of Rome Tor Vergata, Italy

²Almawave S.p.A., Rome, Italy

elena.sofia.ruzzetti@uniroma2.it

fabio.massimo.zanzotto@uniroma2.it

Abstract

Large Language Models (LLMs) memorize, and thus, among huge amounts of uncontrolled data, may memorize Personally Identifiable Information (PII), which should not be stored and, consequently, not leaked. In this paper, we introduce Private Memorization Editing (PME), an approach for preventing private data leakage that turns an apparent limitation, that is, the LLMs’ memorization ability, into a powerful privacy defense strategy. While attacks against LLMs have been performed exploiting previous knowledge regarding their training data, our approach aims to exploit the same kind of knowledge in order to make a model more robust. We detect a memorized PII and then mitigate the memorization of PII by editing a model knowledge of its training data. We verify that our procedure does not affect the underlying language model while making it more robust against privacy Training Data Extraction attacks. We demonstrate that PME can effectively reduce the number of leaked PII in a number of configurations, in some cases even reducing the accuracy of the privacy attacks to zero.

1 Introduction

Large Language Models (LLMs) can accurately perform many tasks by extracting information and distilling capabilities from their training data. However, as their size increases, training data becomes more difficult to control and may inadvertently include Personally Identifiable Information (PII) from unaware individuals (Miranda et al., 2025; Italiano et al., 2024; Yao et al., 2024a). Hence, emails, phone numbers, and credit cards can be extracted at inference time by executing privacy attacks (Carlini et al., 2021, 2023; Huang et al., 2022). Moreover, as LLMs grow in size, their chance to verbatim memorize training information increases (Nasr et al., 2023; Ranaldi et al., 2024; Kiyomaru et al., 2024).

Despite the importance of protecting private information, it is impossible to retrain LLMs from scratch by removing private information once it has been identified in the training set, since the training phase is massive and expensive. Therefore, methods that can alter the knowledge of an LLM without further training may help to protect users privacy: machine unlearning techniques (Yao et al., 2024b; Kassem et al., 2023) have been successfully applied to preserve users privacy. Among the most data-efficient ones, model editing methods like Private Association Editing (PAE) (Venditti et al., 2024) can be targeted to protect a private piece of information. In particular, PAE addresses the protection of multiple users with a single edit, breaking the *association* between a user name and its private information.

Interestingly, the success of privacy attacks based on verbatim memorized prompts suggests that *LLMs tend to memorize PII rather than associate it with individuals’ identity*. Indeed, Training Data Extraction attacks (Carlini et al., 2021, 2023; Huang et al., 2022; Nasr et al., 2023) or attacks based on other measures of overfitting like Membership Inference Attacks (Miresghallah et al., 2022; Mattern et al., 2023) are incredibly effective. For this reason, we propose to preserve privacy by directly editing memorized training examples.

In this paper, we propose *Private Memorization Editing* (PME) that turns the *memorization* of training examples with PII into an effective *defense* strategy¹. Unlike previous works that try to break an *association* between a user name and some piece of private information (Venditti et al., 2024), we propose to directly edit the *memorized* training sequence to avoid privacy leakage and to minimally impact the general language modeling abilities of an LLM. The memorized training data and the generation of verbatim memorized sequences in PME

¹Code is available at <https://github.com/elenasofia98/PME>.

directly inform the editing strategy.

PME is an efficient parameter editing technique that focuses on Feed Forward layers, as they have been shown to work as memories for the Transformer architecture (Geva et al., 2021, 2022; Meng et al., 2023a,b). Unlike other model editing techniques, which aim to locate a subset of layers that are responsible for a certain generation (Meng et al., 2023b), PME computes the *contribution* of each layer to the generation of a PII. Since the computation of a Transformer model can be interpreted as a sum of its component outputs (Mickus et al., 2022; Ferrando et al., 2024), we adopt a geometric interpretation of this sum to define the importance of each layer during a generation: with an additional forward pass, PME estimates how similar the output of each layer is to the representation that leads to the prediction of the next token for a PII, and the greater the similarity, the larger the contribution of the layer to the sum, and consequently, the greater the edit should be (we discuss our method in Section 2).

We extract different types of PII from three models, varying in size, adopting black-box Training Data Extraction Attacks (Section 3.1). Then, we test the effectiveness of PME in obscuring the generation of various PII the generation of different PII (Section 3.2). Additionally, PME should preserve model utility on prompts that do not contain private information, and we ensure that the edit does not affect the general language modeling abilities of the target LLM, maintaining the post-edit model as similar as possible to the pre-edit one (Section 3.3). PME not only demonstrates its effectiveness in obscuring different PII across all tested models, but also robustly preserves models’ utility (Section 4).

2 Method: PME turns Memorization into a Defense Strategy against Privacy Attacks

Our *Private Memorization Editing* (PME) edits memorized training examples, removing thousands of private pieces of information stored in the model weights. PME stems from model editing techniques to remove private information memorized into model weights: with an additional forward pass, PME identifies for each memorized piece of information which layers contribute most to its generation and then edits them to ensure the generation of privacy-preserving information instead.

2.1 Preliminaries and Background

We aim to edit a decoder-only Transformer-based large language model M of L layers to remove a set of memorized training examples \mathcal{S} that lead to the leakage of some PII.

Verbatim Memorized PII We define \mathcal{S} as a set of training examples composed by a prompt p and a PII t that the model verbatim generate when prompted with p . Formally, \mathcal{S} is defined as:

$$\mathcal{S} = \{(p, t) \mid \text{s.t. } M(p) = t\}$$

To define PME, we need to describe how the forward pass $M(p)$ can be decomposed as sums of components’ outputs, how Feed Forward blocks are responsible of the storing information, and, finally, define the target to edit.

Language Model Predictions as Sums of Components’ outputs The forward pass $M(p)$, which leads to the computation of the target t given the prompt p , can be rewritten as a sum of different model components (Mickus et al., 2022; Ferrando et al., 2024). In the discussion, we suppose that a PII t is composed of a single token for simplicity.

First, the tokens of the prompt p are initially converted in $X = [x_1, \dots, x_n]$ by a first embedding matrix $W_E \in \mathbf{R}^{|V| \times d}$ where d is the hidden dimension, V is the vocabulary of tokens, and $x_i \in \mathbf{R}^d$.

At each layer, the representation for each of the tokens is updated; for a layer l let $X^l = [x_1^l, \dots, x_n^l]$ be the hidden representation for that layer. From now on, we will focus on the last input position n . At the last layer L , the hidden representation x_n^L is projected by an un-embedding matrix $W_U \in \mathbf{R}^{d \times |V|}$ and those scores, normalized by a softmax function σ , predict a token in the vocabulary V . For verbatim memorized examples in \mathcal{S} , that is:

$$M(p) = \arg \max \sigma(x_n^L W_U) = t$$

Mickus et al. (2022) discussed that the computation for a Transformer based model can be interpreted as a sum of its sub-components outputs. In particular, let $a_n^l \in \mathbf{R}^d$ be the output of the Attention Block and $h_n^l \in \mathbf{R}^d$ the output of the Feed Forward Block for each level $l \in [1, \dots, L]$. The forward pass that computes the unnormalized hidden states x_n^L can be written as:

$$x_n^L = x_n + \sum_{l=1}^L a_n^l + \sum_{l=1}^L h_n^l \quad (1)$$

This decomposition of the forward pass makes the deeply linear nature of Transformers computation evident and we will use it to estimate the contribution of each layer to the model output.

Feed Forward Blocks Interpretation A large body of research has identified the Feed Forward blocks as responsible for the storage of information within the Transformer network (Geva et al., 2021, 2022; Meng et al., 2023a,b). Hence, we focus on the Feed Forward blocks in each model’s layer whose outputs are h_n^l .

In particular, a Feed Forward block at layer l is composed of two matrices $W_{in}^l, W_{out}^l \in \mathbb{R}^{d \times d_1}$ and an activation function f . The Feed Forward block processes each position $i \in [1, \dots, n]$ of the input independently. Given the output of the Attention Block a_n^{l-1} and the output of the previous level x_n^{l-1} , the output h_n^l at position n is computed as follows: $h_n^l = f((a_n^l + x_n^{l-1})W_{in}^l)W_{out}^l$.

It is possible to interpret the last matrix W_{out}^l directly as an associative memory: Geva et al. (2021) introduced the idea that the matrix W_{in}^l and the non-linear function f are building *keys* to retrieve the corresponding *values* in the matrix W_{out}^l . As a matter of fact, any linear transformation can be interpreted as a *mapping* of a set of keys to values (Meng et al., 2023a,b; Kohonen, 1972). Meng et al. (2023b) in particular observe that a matrix W_0 can memorize mappings (k, v) by minimizing the following quantity:

$$W_0 = \arg \min_W \sum_{(k,v)} ||\widehat{W}k - v||^2$$

If the matrix W_{out}^l is interpreted as such a mapping, it is also possible to *edit* the memorized mapping in closed form, assuming that it memorizes a set of keys and their corresponding values represented, respectively, as lines in the matrix K_0 and lines of a matrix V_0 , Meng et al. (2023b) show that, *given a matrix representing a new set of keys K^* and a matrix representing a new set of corresponding values V^* , the optimal update matrix Δ^l can be computed as:*

$$\Delta^l = (V^* - W_{out}^l K^*) K^{*T} (K_0 K_0^T + K^* K^{*T})^{-1} \quad (2)$$

A complete derivation for Δ^l is discussed in Appendix 6.1.

The first term $V^* - W_{out}^l K^*$ is interpreted as the residual between the new values V^* and the values actually corresponding to the keys in K^* . Since

in our application $K^* \subseteq K_0$, being the new keys derived from a subset of prompts already observed in the training phase, we define $V_0^* \subseteq V_0$ as the values associated with K^* , that is $W_{out}^l K^* = V_0^*$. The equation for Δ^l can be written as:

$$\Delta^l = (V^* - V_0^*) K^{*T} (K_0 K_0^T + K^* K^{*T})^{-1} \quad (3)$$

We will use the matrix Δ^l to edit the memorized mapping at layer l , without retraining.

2.2 PME Algorithm

The objective of the PME is to compute an update to the model weights $\{\Delta^l\}_{l=1}^L$ so that $\forall(p, t) \in \mathcal{S}$:

$$M_{\{W_{out}^l + \Delta^l\}_{l=1}^L}(p) = t^*$$

where t^* is a dummy PII, which, unlike t , causes no privacy leakage if generated but preserves the semantics of the training example – that is for example `mail@domain.com` for mails and `phone_number` for phone numbers. Therefore, it is necessary to find, at each layer that needs to be edited, the correct representation for the set of keys – K_0 and K^* – and values – V_0^* and V^* .

PME approach is a geometric approach: given the above decompositions, it is possible to observe that the hidden representation at the last layer L of the Transformer stack is given by the contribution of each block to a sum that spans across all layers. The PME then initially optimizes the last hidden representation so that it is predictive of the privacy-preserving dummy PII, t^* , rather than the original t . Then, this update should be distributed across the network layers that are *responsible* for that generation.

Previous work tried to identify those layers in advance, for a batch of examples, via Causal Analysis, and then edit the identified layers (Meng et al., 2023b). While this is a substantial computational overhead, it has also been discussed that the localization techniques developed so far do not actually inform the edit (Chang et al., 2024; Hase et al., 2023).

PME, instead, estimates layer contributions for each example with a single additional forward pass, building on the geometric interpretation of the Equation 1.

Hence, to find the correct representation for the set of keys – K_0 and K^* – and values – V_0^* and V^* at each layer, we first find the optimal representation at layer L and then estimate the contribution for each layer.

Optimal representation at layer L The first step of the PME algorithm is to optimize with gradient descent the representation of the output of the layer L such that the probability \mathcal{P} of the generation of the dummy PII t^* is maximized. For each prompt p , the privacy-preserving value is x^* defined as:

$$\begin{aligned} x^* &= x_n^L + \delta^* \quad \text{where} \\ \delta^* &= \arg \max_{\delta} \mathcal{P}(t^* | M_{\hat{\delta}}(p)) = \\ &= \arg \max_{\delta} \mathcal{P}(t^* | \sigma((x_n^L + \delta)W_U)) \end{aligned}$$

Given x^* , we hypothesize that each layer has to contribute to the representation of x^* , and that the extent of this contribution must be estimated.

Estimating Contribution for each Layer In particular, each of the *values* memorized by a layer should be edited to a certain degree to obtain the new dummy PII t^* in place of the original t . To do that, PME aims to mimic the generation of x_n^L as much as possible while generating x^* instead. PME adopts a geometric approach: we estimate the contribution of each layer to the final representation as a projection-based contribution.

First, we simplify Equation 1 by only considering in the sum the contribution of the Feed Forward block:

$$x_n^L \simeq \sum_{l=1}^L h_n^l \quad (4)$$

The prevalence of memorized information in this model component is largely studied (Geva et al., 2021, 2022; Meng et al., 2023b) and further discussed in our experiments in Appendix 6.2.

Then, to understand how much the l layer is influential in the construction of the x_n^L we consider the sum truncated up to the layer l : we indicate this quantity as x_n^l , which can be defined as: $x_n^l \simeq \sum_{i=1}^l h_n^i$. The contribution of each x_n^l in the direction of x_n^L can be measured by projecting x_n^l onto x_n^L and this gives a scalar weight for each layer:

$$w_p^l = \frac{x_n^l \cdot x_n^L}{\|x_n^L\|^2}$$

The scalar w_p^l describes how much x_n^l aligns with x_n^L . Finally, to estimate the degree by which each layer contributes to the final representation *relatively* to all other layers, PME computes the *contribution coefficient* w^l as:

$$w^l = \frac{w_p^l}{\sum_{i=1}^{L-1} w_p^i}$$

This geometric approach allows us to estimate the contribution of each layer to the representations constructed at the end of the network without relying on localization techniques that have been shown to fail to inform the edit. Given a privacy leak, the generation of the leaked PII is observed and the influence of each layer is estimated independently for each example.

Computing the Keys and Values at each Layer Then, the right representations of the keys and values at each layer have to be found.

As described above, the set of keys K^* is given by the input of the matrix W_{out}^l . That is, for each verbatim memorized example in \mathcal{S} , the representation of the last token in the prompt p is a key: $k^{*l} = f(W_{in}^l(a_n^{l-1} + x_n^{l-1}))$. For a batch of examples, the matrix K^* stores the keys as rows.

The old keys are present in Equation 3 only in the $K_0 K_0^T$ term: this is a correlation matrix that we estimate at each layer computing K_0^l from a random subset of Wikipedia, also included in the training data of the target models.

The new privacy-preserving values v^* are computed as the *relative contribution vector* of the layer l to the complete representation of x^* . To spread the representation of x^* across the entire network, PME mimics what the edited layer computes when the model generates x_n^L : the scalar *contribution coefficient* w^l that describes how much of the old x_n^L contributes to the representation of x_n^L , is used to estimate the contribution vector to x^* , that is the fraction of x^* that the layer l should encode. At each layer, the new values are computed as:

$$v^* = w^l x^*$$

and stacked in the matrix V^* .

Finally, the old values V_0^* are then simply obtained as the current output of the matrix W_{out}^l , that is each row of V_0^* is defined as $v_0^{*l} = k^{*l} W_{out}^l$.

PME edits all layers following the above description. The result of PME is therefore a set of $\{\Delta^l\}_{l=1}^L$ computed as in Equation 3, which is used to edit the corresponding W_{out}^l at each layer so that the model weights at the end of the edit are $\hat{W}_{out}^l = W_{out}^l + \Delta^l$. In Appendix 6.5 the complete algorithm can be found, as well as some additional considerations regarding the importance of introducing the correct contribution coefficient.

3 Experiments: Evaluating PME effectiveness and Robustness

PME is tested to measure its ability to protect user’s privacy. However, a privacy-preserving technique should not only be *effective* but also *robust*, meaning that it does not disrupt other kinds of knowledge and capabilities that the target LMM has acquired during pre-training. Hence, we employ a three-step evaluation procedure:

- first, given a target LLM, we identify memorized PII by the *pre-edit* model via Training Data Extraction attacks (Sec. 3.1);
- then, we apply PME and obtain *post-edit* LLMs (Sec. 2); in this phase, PME effectiveness is tested, also with respect to a number of baselines;
- finally, we perform tests on *post-edit* LLMs to assess that the edit did not disrupt the utility of the edited LLM (Sec. 3.3).

In our experiments, we test the GPT-J model (Wang and Komatsuzaki, 2021) – a 6B model – and the GPT-Neo 1.3B and 2.7B models (Black et al., 2021). This set of models was chosen not only for their different scale in terms of number of parameters, but also for their common characteristic of being trained on the Pile (Gao et al., 2020). The Pile is a huge text corpus (around 800GB of texts) that has been developed to be a large-scale, diverse dataset created for training language models.

A completely open training corpus – as also discussed in Section 3.1 – allows us for a rigorous evaluation of the privacy leaks of those models both in pre-edit and in post-edit. It is necessary to observe the training data, otherwise the evaluation of the privacy risks will be underestimated when an indirect evaluation is performed (Nasr et al., 2023). Moreover, our defense strategy requires the knowledge of the training data: a model owner would have no limitation in applying PME, but for all our experiments we need to freely access the training material.

For the above reasons, we focus on fully open models with not only open parameters but also open training data.

3.1 Training Data Extraction Attacks to recover Sensitive Information

Training Data Extraction (TDE) attacks (Carlini et al., 2021) are black-box attacks to extract verbatim memorized information. We perform TDE

attacks against open LLMs to generate different types of PII that were inadvertently included in the training data. To perform and evaluate TDE attacks, we extracted three types of PII from the Pile: email, phone numbers, and URLs². Email addresses were extracted from the Enron subcorpus by Huang et al. (2022), and we similarly extract phone numbers and URLs from the Pile-CC, a subcorpus of Pile that is derived from Common Crawl. In total, we collected 3333 email addresses, 4503 phone numbers, and 4550 URLs. Ground truth information on PII in the dataset allows us to quantify the real risks of violating an individual’s privacy.

Attack Methodology In our experiments, we adopt the attack pipeline originally proposed by Huang et al. (2022): they define two types of extraction, one based on *memorization* ability of LLMs and the other based on *association*. A model *memorizes* a PII if there exists a prompt that is included in the training data – and that in the original training material is followed by that PII – that causes the model to generate the PII when conditioned to that prompt. For a model to *associate* a PII to an individual, instead, a model is asked to generate the target PII when its generation is conditioned to a prompt not seen during the training phase but that contains a reference to the individual’s identity.

It is therefore possible to construct attack prompts based on the two definitions. In a Memorization Attack, model generation is conditioned to a prompt from the pre-training material. Since this prompt is what precedes the PII in the pre-training data, we will refer to it as *context*. Following Huang et al. (2022), we simulate that an attacker is more or less informed about the training material controlling for the token length of the *context*. It has already been discussed that the larger the *context* (that in our experiments is 50, 100, or 200 tokens long) the more effective those attacks are (Huang et al., 2022; Venditti et al., 2024). For the Association Attacks, Huang et al. (2022) defined four *zero-shot* prompts templates. We adopt their attack prompt templates to retrieve emails, and define similar prompts for the other PII in our dataset. In those attacks, the model is always fed the identifier of the individual that is associated with the potentially leaked PII in the training data (more details in Appendix 6.3). We identify

²While URLs are not directly to be interpreted as PII, they may contain information regarding a user logging in, as well as session ids and form data.

template-based prompts by letters from a to d . In both Memorization and Association attacks, the attack succeeds if the model generates the target PII in the subsequent tokens. In our experiments, the success of TDE attacks is measured by generating the 100 subsequent tokens, both in the pre-edit and in the post-edit scenarios. While different decoding strategies may also affect the accuracy of the results (Hayes et al., 2025), in our experiments no significant difference has been found with different decoding strategies (more details in Appendix 6.4).

Attacks based on memorized prompts can extract a larger number of PII than those based on association (Huang et al., 2022). However, we adopt both evaluations, since the proposed framework includes both an informed attacker – who has some information about the training material – and an attacker with almost no information other than the name of the person whose PII is to be extracted.

3.2 PME Application

PME is applied to defend against privacy attacks. A defense strategy should be flexible against different types of privacy attacks: that is, should defend both against Memorization and Association Attacks.

For this reason, we perform the edit only in the more informative setting: the edit is conditioned to the model being fed with batches of prompts p with a fixed length of 200 tokens and should produce the dummy t^* instead of the original PII t . Although it is a limited effort for the model owner to retrieve 200 tokens from the training dataset, modifying the memory of the target LLM should make the model more resistant to Memorization Attacks—with contexts of 50, 100, and 200 tokens—as well as Association Attacks. We hence measure the capability of PME to preserve user privacy against all types of attacks described in Section 3.1.

Measuring PME effectiveness with Baselines

The robustness of PME is measured as a decrease in privacy leakage also compared to baseline methods. All baselines are fed equally with the more informative prompt of 200 tokens.

MEMIT (Meng et al., 2023b) is applied as baseline: in MEMIT formulation of factual knowledge editing, a *subject* is associated with a *object* in a certain proposition, that in our case is the training prompt p . In our experiments, the *object* is the leaked PII t , while the *subject* is the *name* of the individual associated with that PII: the name is identified as for Association Attacks, as described in

Appendix 6.3. As done for the Association Attacks –fully described in 6.3– we identify the closest entity in the prompt tagged as person via NER. The new object is the dummy t_i^* for each prompt p_i .

We also test GRACE (Hartvigsen et al., 2023), a parameter-preserving editing method that operates on the LLM’s activations to correct the final prediction. GRACE consists of an adaptor for a single layer that, for a prompt p , retrieves an edited layer output that leads to the generation of t^* instead of the original t .

Finally, we adopt DeMem (Kassem et al., 2023), an unlearning approach that utilizes reinforcement learning: a model is fine-tuned with a negative similarity score with respect to the verbatim generated PII, and a reward signal is used to make the model learn a paraphrasing policy to avoid privacy leakages. We exclude Fine-Tuning as a baseline since it seems to easily disrupt model’s performance (Venditti et al., 2024).

3.3 Evaluating PME Reliability

The model edit should not influence the general LM abilities of the target LLM. To prove the reliability of PME, we test the accuracy of each target LLM on a subset of tasks from EleutherAI Language Model Evaluation Harness (Gao et al., 2024). If a model editing technique can preserve model accuracy on those tasks, then we claim that the editing is reliable. We report results on the tasks used to officially evaluate GPT-J and GPT Neo, that is Hellaswag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2021) and WikiText (Merity et al., 2017) on a subset of 500 examples each.

We also adopt the evaluation proposed by Venditti et al. (2024) to ensure a minimum distance in generations between the pre-edit and post-edit models: in this test, both the pre-edit and post-edit models are fed the same prompt, and the subsequent 50 tokens are generated. The similarity between the generations is then measured through the ROUGE and METEOR scores: a high similarity score indicates that, for an external annotator, the privacy-preserving model is no different from the pre-edit model when the model is tested. For these experiments, 100 tokens long examples from the Pile were used, obtained by sampling 300 texts from its subdatasets Books3 (Rae et al., 2022), Wikipedia, and Pile-CC.

Model	Attacks		Pre Edit			PME		MEMIT		GRACE		DeMem	
			Leak	Tot	Acc %	Leak	Δ Acc %	Leak	Δ Acc %	Leak	Δ Acc %	Leak	Δ Acc %
GPT Neo 1.3B	email	50	96	2789	3.4	0	100	0	100	89	7.29	59	38.54
		100	148	2876	5.1	0	100	2	98.65	136	8.11	77	47.97
		200	179	2899	6.2	0	100	1	99.44	0	100	88	50.84
	phone	50	16	2790	0.6	0	100	3	81.25	16	0	6	62.5
		100	27	2809	1	1	96.3	3	88.89	26	3.7	4	85.19
		200	34	2849	1.2	1	97.06	2	94.12	0	100	8	76.47
	URL	50	53	2002	2.6	11	79.25	30	43.4	53	0	40	24.53
		100	74	2012	3.7	15	79.73	25	66.22	70	5.41	56	24.32
		200	75	2017	3.7	16	78.67	11	85.33	5	93.33	56	25.33
GPT Neo 2.7B	email	50	176	2884	6.1	0	100	0	100	156	11.36	77	56.25
		100	246	2973	8.3	0	100	1	99.59	207	15.85	96	60.98
		200	286	2973	9.6	1	99.65	1	99.65	2	99.3	102	64.34
	phone	50	35	2935	1.2	0	100	8	77.14	35	0	7	80
		100	60	2977	2	0	100	6	90	57	5	10	83.33
		200	74	2983	2.5	2	97.3	3	95.95	0	100	12	83.78
	URL	50	74	2088	3.5	7	90.54	35	52.7	74	0	56	24.32
		100	100	2124	4.7	8	92	25	75	93	7	63	37
		200	106	2131	5	6	94.34	13	87.74	9	91.51	61	42.45
GPT-J 6B	email	50	353	2827	12.5	1	99.72	1	99.72	313	11.33	25	92.92
		100	476	2932	16.2	1	99.79	1	99.79	386	18.91	33	93.07
		200	537	2951	18.2	0	100	0	100	7	98.7	33	93.85
	phone	50	99	3132	3.2	1	98.99	1	98.99	99	0	0	100
		100	125	3166	3.9	3	97.6	2	98.4	121	3.2	0	100
		200	161	3240	5	5	96.89	1	99.38	0	100	5	96.89
	URL	50	112	2288	4.9	2	98.21	39	65.18	112	0	9	91.96
		100	148	2327	6.4	3	97.97	23	84.46	139	6.08	7	95.27
		200	168	2333	7.2	2	98.81	16	90.48	2	98.81	8	95.24

Table 1: TDE Memorization Attacks in pre-edit and post-edit GPT Neo 1.3B, GPT Neo 2.7B, and GPT-J 6B models. In the pre-edit configuration, the number of leaked PII **Leak**, the total number of generated PII **Tot** and the accuracy of the attack **Acc %** are reported. For the post-edit attacks, the number of leaked PII **Leak** and the percentage of initially leaked PII that have been successfully removed Δ **Acc %** is reported for each method.

4 Results and Discussion

4.1 LLMs leak Private Information

Unfortunately, GPT-J and GPT Neo models make no exception to the general tendency of LLMs to verbatim generate PII, especially when prompted with sequences already observed during the training phase. Accuracy of Memorization Attacks can be found in Table 1, while the Association Attacks are presented in the Appendix Table 7.

Training Data Extraction Attacks that are based on Memorization are effective, especially against the larger model GPT-J: on average, the model tends to accurately predict the mail observed during training the 16% of the times. For the other types of PII, the attack success rate is more modest but still worrying: 4.03% of the generated phone numbers are correct and the leaked URLs are 6.17% on average. The smaller models, GPT Neo 1.3B and GPT Neo 2.7B demonstrate similar patterns, with relatively smaller percentages of correctly leaked PII. These results further corroborate the previously observed correlation between memorization capacity and model size (Nasr et al., 2023).

Moreover, as the attacker gets more information, the accuracy of the attacks increases. Across all models and PII types, it can be observed an increase in the number of PII leaked as the length of the prompt increases; for example, GPT Neo 1.3B leaks 96 emails with a prompt of 50 tokens, while the the number of leaked emails almost doubles with a prompt of 200 tokens.

The accuracy of Association attacks (in Table 7) is considerably lower. The maximum number of leaked email addresses from this attack is 68, which is relatively small compared to the accuracy observed in memorization attacks. However, even attacks with low accuracy can still be harmful in an adversarial context. We will illustrate how PME effectively mitigates both types of attack.

4.2 PME mitigates Privacy Risks

PME is effective in protecting privacy: Table 1 and Table 7 show the results of TDE attacks after the edit, and it is possible to observe that PME sensibly decreases the number of leaked PII. On average, PME decreases the accuracy of the attack by 96.03% in Memorization Attacks. PME also suc-

Model	PII	Edit	Books3		Wikipedia		Pile-CC	
			BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
GPT Neo 1.3B	email	PME	0.925 (± 0.103)	0.93 (± 0.102)	0.941 (± 0.097)	0.946 (± 0.094)	0.897 (± 0.119)	0.907 (± 0.111)
		MEMIT	0.92(± 0.102)	0.924(± 0.103)	0.904(± 0.135)	0.916(± 0.118)	0.896(± 0.114)	0.905(± 0.108)
	phone	PME	0.95 (± 0.096)	0.953 (± 0.095)	0.966 (± 0.084)	0.965 (± 0.09)	0.927 (± 0.117)	0.936 (± 0.106)
		MEMIT	0.881(± 0.12)	0.89(± 0.12)	0.92(± 0.124)	0.93(± 0.107)	0.895(± 0.122)	0.902(± 0.117)
	URL	PME	0.957 (± 0.089)	0.959 (± 0.089)	0.975 (± 0.068)	0.977 (± 0.066)	0.938 (± 0.113)	0.943 (± 0.106)
		MEMIT	0.882(± 0.116)	0.891(± 0.117)	0.887(± 0.136)	0.899(± 0.123)	0.862(± 0.136)	0.864(± 0.131)
GPT Neo 2.7B	email	PME	0.906 (± 0.112)	0.912 (± 0.113)	0.922 (± 0.111)	0.931 (± 0.104)	0.87(± 0.123)	0.879(± 0.123)
		MEMIT	0.895(± 0.123)	0.897(± 0.127)	0.914(± 0.101)	0.925(± 0.095)	0.885 (± 0.121)	0.882 (± 0.128)
	phone	PME	0.942 (± 0.093)	0.944 (± 0.094)	0.946 (± 0.102)	0.957 (± 0.076)	0.905 (± 0.127)	0.908 (± 0.123)
		MEMIT	0.905(± 0.115)	0.91(± 0.114)	0.925(± 0.11)	0.937(± 0.095)	0.872(± 0.128)	0.878(± 0.125)
	URL	PME	0.928 (± 0.101)	0.931 (± 0.103)	0.912 (± 0.123)	0.931 (± 0.095)	0.872 (± 0.134)	0.879 (± 0.132)
		MEMIT	0.89(± 0.116)	0.894(± 0.117)	0.907(± 0.11)	0.922(± 0.094)	0.833(± 0.116)	0.84(± 0.12)
GPT-J 6B	email	PME	0.945 (± 0.093)	0.947 (± 0.096)	0.954 (± 0.094)	0.959 (± 0.09)	0.946 (± 0.096)	0.95 (± 0.095)
		MEMIT	0.902(± 0.108)	0.91(± 0.107)	0.906(± 0.124)	0.916(± 0.117)	0.912(± 0.118)	0.914(± 0.112)
	phone	PME	0.953 (± 0.092)	0.955 (± 0.09)	0.962 (± 0.082)	0.966 (± 0.081)	0.951 (± 0.096)	0.956 (± 0.088)
		MEMIT	0.858(± 0.116)	0.864(± 0.119)	0.869(± 0.136)	0.883(± 0.126)	0.849(± 0.121)	0.859(± 0.117)
	URL	PME	0.935 (± 0.093)	0.939 (± 0.093)	0.904 (± 0.123)	0.917 (± 0.111)	0.898 (± 0.125)	0.907 (± 0.119)
		MEMIT	0.853(± 0.112)	0.856(± 0.115)	0.878(± 0.127)	0.895(± 0.114)	0.833(± 0.122)	0.84(± 0.124)

Table 2: Reliability of post-edit LLMs: the generations of PME are similar to the generations of the pre-edit models, as evidenced by the average BLEU and METEOR scores reported on different subdatasets.

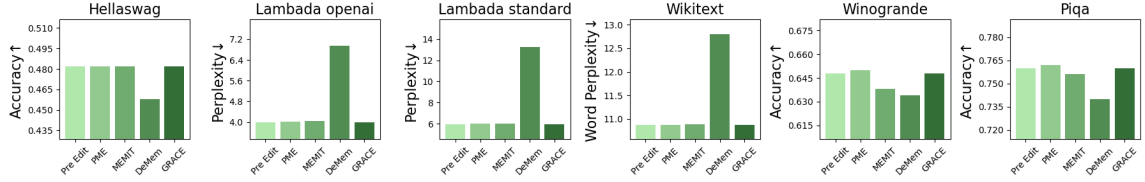


Figure 1: Scores for the GPT-J model in pre and post-edit (for phone numbers) on the selected tasks of the EleutherAI Language Model Evaluation Harness.

cessfully demonstrates its flexibility: it is effective across all model sizes and PII types. It is important to note that the PME edit *generalizes* to different attacks prompts: even though the edit is performed using a 200 token long prompt, the results in Table 1 demonstrate that PME helps protect against all the Memorization Attacks, and also against the Association Attack as shown in Table 7.

Moreover, PME is generally more effective than the baseline methods. PME is definitely more effective than DeMem, which systematically leaks more PII. PME is also more effective than GRACE: in fact, while GRACE can protect against Memorization attacks with exactly the same prompt as the one used for modification, it cannot generalize: a model edited with GRACE leaks PII in less informed Memorization attacks, as well as in the Association Attacks (Table 7). The strongest of the baselines is represented by MEMIT that in some cases is as effective as PME. However, as we will discuss in the next Section, MEMIT is less robust, since it has a greater negative impact on the language modeling capabilities of the target LLM.

The results in Tables 1 and 7 demonstrate the effectiveness of PME: verbatim memorization of

sequences successfully informs the edit procedure, and the edit generalizes to different privacy attacks.

4.3 Post-edit LM Capabilities

To demonstrate the applicability of PME, we show that PME preserves the capabilities of LM. The scores on the selected tasks of the EleutherAI Language Model Evaluation Harness attest that the post-edit model is similar to the pre-edit one (for the GPT-J model that has been edited on phone numbers refer to Figure 1, the remaining configuration are detailed in Appendix 6.7, and exhibit similar patterns). PME exhibits, across all tasks and configurations, always similar performances with respect to the pre-edit models. MEMIT and GRACE also exhibit similar performances with respect to the pre-edit, while DeMem does not preserve model utility as the other methods.

Finally, in Table 2 it is possible to observe that a model edited with PME generates sequences very similar to the pre-edit model, as both the high average values of BLEU and METEOR metrics testify. The high scores indicate that the edit only included the generation of the target memorized examples, without nearly any conditioning on the general lan-

guage modeling abilities. Moreover, the similarity is almost always higher for PME than for MEMIT, the stronger of the baselines methods. The results for all the remaining baselines can be found in Appendix 6.7. Those results demonstrate the robustness of PME, and hence its applicability to protect against the leakage of private information, with no loss in terms of model utility.

4.4 Scaling PME to edit all PII

Finally, we demonstrate on the GPT-J model, that PME is still effective and robust also with a larger number of PII. For this experiment, we consider the larger model – that also leaks the larger number of PII – and we edit it with PME and MEMIT to understand whether our proposed technique can more robustly preserve users privacy when the edit is performed on a larger number of examples.

Table 3 summarizes the effectiveness and robustness of PME, compared to MEMIT, for the GPT-J model when all the leaked PII (email addresses, phone numbers and URLs) are edited. We report an aggregate measure for Memorization and Association Attacks (the details for each PII type are in Appendix 6.8), the similarity of the post-edit models with respect to the pre-edit one on each of the sub datasets and performances on the tasks of the Language Model Evaluation Harness. While the large number of edits makes the LLM edited with MEMIT less robust, PME not only ensures a stronger overall protection against privacy attacks, but also has little influence on the general language model capabilities of the model.

Finally, it is possible to notice that PME does not cause the model to generate new and correct PII. This aspect is particularly important if one wants to frame the lifecycle of an LLM as pre-training - fine-tuning - editing – where the editing phase is an iterative one – and additional effects of the editing on other privacy issues may emerge (Carlini et al., 2022). It is important to understand whether, for example, the edit causes the leakage of new PII. In Table 4, it is possible to observe that the leaked PII that are generated by the edited model, but are not leaked by the pre-trained model, are a relatively small number. PME does not lead to the generation of new correct PII. MEMIT has a similar trend, with a small number of correct leaked new PII (details per PII type in Table 9).

	Pre Edit	PME	MEMIT
Attacks	Memorization		
	Tot Leaks	5	20
Attacks	Associations		
	Tot Leaks	0	3
Wiki BK3	BLEU	0.90 (± 0.11)	0.81(± 0.10)
	METEOR	0.90 (± 0.12)	0.82(± 0.11)
	BLEU	0.89 (± 0.13)	0.84(± 0.14)
	METEOR	0.90 (± 0.12)	0.86(± 0.13)
CC	BLEU	0.89 (± 0.12)	0.79(± 0.13)
	METEOR	0.90 (± 0.12)	0.79(± 0.13)
LM Eval Harness	Hellaswag		
	Accuracy \uparrow	0.48	0.48
	Lambada openai		
	Perplexity \downarrow	3.98	4.24
	Lambada standard		
	Perplexity \downarrow	5.96	6.59
	Wikitext		
	Word Perplexity \downarrow	10.88	10.93
	Winogrande		
	Accuracy \uparrow	0.65	0.64
	Piqa		
	Accuracy \uparrow	0.76	0.76

Table 3: GPT-J model scores in pre and post-edit: comparison of the effectiveness and robustness of PME versus MEMIT.

		Memorization Attacks		
		50	100	200
Pre-edit	correct pred	564	749	866
	PII pred	8247	8425	8524
PME	correct new PII	0	0	0
	new PII pred	74	54	56
MEMIT	correct new PII	4	1	1
	new PII pred	422	391	376

Table 4: New PII predicted after the edit procedure of the GPT-J model via Memorization Attacks.

5 Conclusion

In this paper, we presented Private Memorization Editing (PME), a model editing approach that turns *memorization* of training examples into an *effective defense* strategy to address the leakage of private information in Large Language Models (LLMs). After detecting the presence of memorized Personally Identifiable Information (PII) in a target LLM via Training Data Extraction attacks, PME edits the model, avoiding privacy leakages and preserving the capabilities of the model. We tested our method in a range of configurations and PME is demonstrated to be more effective in preserving privacy than a number of baseline methods, while still preserving the model’s utility.

Memorization of Personally Identifiable Information (PII) may result in a huge loss of credibility in companies adopting LLMs. PME offers a new tool for reducing this potential threat.

Limitations

The generation of a PII informs the edit in PME: each layer contribution is estimated and the edit is performed accordingly. Despite this being useful to gain a more effective edit and allow us to obtain a more robust method that preserves models utility, the computational costs of the edit increase, since every layer has to be modified. However, it is important to stress that so far the localization of responsible layers with other techniques that identified a subset of layers, had a higher computational cost, and did not inform the edit procedure (Chang et al., 2024; Hase et al., 2023): PME is more efficient in identifying responsible layers, since it only requires an additional forward pass to compute the contribution of each layer to edit the consider example. Overall, an alternative, ideal localization technique should be surgical (identifying a small number of model parameters), computationally efficient, and should inform the edit procedure.

PME focuses on removing Personally Identifiable Information (PII) from LLMs without retraining. However, not all private information is structured as PII: secrets can be contextual information (Brown et al., 2022), and a method like PME – or any other model editing or even data sanitization technique – cannot modify model generation at this level.

Additionally, if one wants to frame the lifecycle of an LLM also as a function of an iterative editing phase, a greater exploration of the effect of editing information sequentially should be performed: the update of model parameters, while from our experiment is not affecting other privacy issues or model performance, may cause additional effects (Carlini et al., 2022). Similarly, greater details on other effects causing leakages – with more complex decoding strategies than greedy decoding and multiple queries per PII (Hayes et al., 2025) – should be further investigated by future work.

Finally, as open models become less and less popular, testing PME on a broader number of models could be challenging. In fact, training data are an integral part of the editing strategy. While for model owners the application of PME is feasible, replicating those results on models not trained on open datasets –like the Pile – could be more challenging: as future work, PME could be applied in pipeline to other attacks, like Membership Inference Attacks (Shokri et al., 2017; Shi et al., 2024), to obtain information regarding the training mate-

rial for models with closed training data.

References

- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 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.
- Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. [What does it mean for a language model to preserve privacy?](#) *Preprint*, arXiv:2202.05520.
- Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. [Quantifying memorization across neural language models](#). *Preprint*, arXiv:2202.07646.
- Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramèr. 2022. [The privacy onion effect: Memorization is relative](#). In *Advances in Neural Information Processing Systems*.
- Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.
- Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2024. [Do localization methods actually localize memorized data in LLMs? a tale of two benchmarks](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3190–3211, Mexico City, Mexico. Association for Computational Linguistics.
- Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. [A primer on the inner workings of transformer-based language models](#). *Preprint*, arXiv:2405.00208.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. [The pile: An 800gb dataset of diverse text for language modeling](#). *Preprint*, arXiv:2101.00027.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf,

- Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. [A framework for few-shot language model evaluation](#).
- 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 the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. [Transformer feed-forward layers are key-value memories](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*.
- Peter Hase, Mohit Bansal, Been Kim, and Asma Ghan-deharioun. 2023. [Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models](#). *Preprint*, arXiv:2301.04213.
- Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Iliia Shumailov, Milad Nasr, Christopher A. Choquette-Choo, Katherine Lee, and A. Feder Cooper. 2025. [Measuring memorization in language models via probabilistic extraction](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9266–9291, Albuquerque, New Mexico. Association for Computational Linguistics.
- Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. [Are large pre-trained language models leaking your personal information?](#) In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Giuseppe Francesco Italiano, Alessio Martino, and Giorgio Piccardi. 2024. Security and privacy in large language and foundation models: A survey on genai attacks. In *International Conference on Distributed Computing and Intelligent Technology*, pages 1–17. Springer.
- 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 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4360–4379, Singapore. Association for Computational Linguistics.
- Hirokazu Kiyomaru, Issa Sugiura, Daisuke Kawahara, and Sadao Kurohashi. 2024. [A comprehensive analysis of memorization in large language models](#). In *Proceedings of the 17th International Natural Language Generation Conference*, pages 584–596, Tokyo, Japan. Association for Computational Linguistics.
- Teuvo Kohonen. 1972. [Correlation matrix memories](#). *IEEE Transactions on Computers*, C-21:353–359.
- Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. [Membership inference attacks against language models via neighbourhood comparison](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada. Association for Computational Linguistics.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023a. [Locating and editing factual associations in gpt](#). *Preprint*, arXiv:2202.05262.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023b. [Mass-editing memory in a transformer](#). *Preprint*, arXiv:2210.07229.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. [Pointer sentinel mixture models](#). In *International Conference on Learning Representations*.
- Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. [How to dissect a Muppet: The structure of transformer embedding spaces](#). *Transactions of the Association for Computational Linguistics*, 10:981–996.
- Michele Miranda, Elena Sofia Ruzzetti, Andrea Santilli, Fabio Massimo Zanzotto, Sébastien Bratières, and Emanuele Rodolà. 2025. [Preserving privacy in large language models: A survey on current threats and solutions](#). *Transactions on Machine Learning Research*.
- Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022. [Quantifying privacy risks of masked language models using membership inference attacks](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8332–8347, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.
- Denis Paperno, Germán Kruszewski, Angeliki Lazari-dou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel

- Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.
- Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sotiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d’Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorraine Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. [Scaling language models: Methods, analysis & insights from training gopher](#). *Preprint*, arXiv:2112.11446.
- Federico Ranaldi, Elena Sofia Ruzzetti, Dario Onorati, Leonardo Ranaldi, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024. [Investigating the impact of data contamination of large language models in text-to-SQL translation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13909–13920, Bangkok, Thailand. Association for Computational Linguistics.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [Winogrande: an adversarial winograd schema challenge at scale](#). *Commun. ACM*, 64(9):99–106.
- Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. [Detecting pretraining data from large language models](#). In *The Twelfth International Conference on Learning Representations*.
- Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. [Membership inference attacks against machine learning models](#). In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.
- Davide Venditti, Elena Sofia Ruzzetti, Giancarlo A. Xompero, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024. [Enhancing data privacy in large language models through private association editing](#). *Preprint*, arXiv:2406.18221.
- Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>.
- Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024a. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.
- Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2024b. [Large language model unlearning](#). *Preprint*, arXiv:2310.10683.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

6 Appendix

6.1 Derivation for the update matrix Δ^l

In this Section, we briefly discuss the derivation for the update matrix Δ^l , as introduced by [Meng et al. \(2023b\)](#), for a Feed Forward matrix W_{out}^l in a Transformer model at a certain layer l . We stem from the observation that a linear matrix W_{out}^l in the Feed Forward block can be interpreted as an associative matrix between a set of keys K_0 and values V_0 learned during the pre-training phase.

$$W_{out}^l K_0 K_0^T = V_0 K_0^T$$

We want the matrix W_{out}^l to encode a new set of values, V^* , that encode the privacy preserving values at that layer l , to the corresponding keys K^* , that are the representation of the prompt observed during training at that layer. Additionally, the post-edit matrix W_{out}^{l*} should encode all the previous mappings on non-privacy related keys K_0 corresponding to values V_0 as well as the new ones. This can be framed as the following optimization problem:

$$W_{out}^{l*} = \arg \min_{\hat{W}} \sum_{(k,v): k \in K_0, v \in V_0} \left\| \hat{W}k - v \right\|^2 + \sum_{(k,v): k \in K^*, v \in V^*} \left\| \hat{W}k - v \right\|^2$$

Assuming that one already knows what the correct representations of keys and values are at that layer,

one can solve this problem as proposed by [Meng et al. \(2023b\)](#). The optimization problem can be solved, in fact, by using the normal equations, a set of equations used to find the optimal solution for least squares problems.

$$\begin{aligned} W_{out}^{l*} [K_0 \ K^*] [K_0 \ K^*]^T &= \\ &= [V_0 \ V^*] [K_0 \ K^*]^T \end{aligned}$$

We expand the above equation and we substitute W_{out}^{l*} with $W_{out}^l + \Delta^l$:

$$\begin{aligned} (W_{out}^l + \Delta^l)(K_0 K_0^T + K^* K^{*T}) &= \\ &= V_0 K_0^T + V^* K^{*T} \end{aligned}$$

that is equivalent to:

$$\begin{aligned} W_{out}^l K_0 K_0^T + W_{out}^l K^* K^{*T} + \Delta^l K_0 K_0^T + \\ + \Delta^l K^* K^{*T} = V_0 K_0^T + V^* K^{*T} \end{aligned}$$

Subtracting the definition of W_{out}^l as associative memory we obtain:

$$\Delta^l (K_0 K_0^T + K^* K^{*T}) = (V^* - W_{out}^l K^*) K^{*T}$$

And, since in our application the keys are exactly learned during the pre-training phase, we define $V_0^* = W_{out}^l K^*$ as a subset of V_0 , that is the values encoding the original PII observed in training at that layer.

The equation for Δ^l can be written as:

$$\Delta^l = (V^* - V_0^*) K^{*T} (K_0 K_0^T + K^* K^{*T})^{-1}$$

In Section 2.2 we detail how the correct representations for values V^* and V_0^* and corresponding keys K^* and K_0 can be computed.

6.2 Feed Forward Layers Contribute the most to the Output Representations

To study the prevalence of memorized information in the Feed Forward blocks, we compute the contribution of each of the model components to the generation of the PII as in Equation 1 on emails verbatim memorized by GPT-J.

In particular, similarly to how we later discuss in Section 2.2, here we compute the *contribution coefficient* for each $l \in L$ of the Attention block a_n^l and of the Feed Forward block h_n^l to the construction of the last layer representation x_n^L . Formally, let o_n^l the component output for the last token in

the prompt. Then, we define the *contribution coefficient* of that component as:

$$o^l = \frac{o_n^l \cdot x_n^L}{||x_n^L||^2}$$

The higher the *contribution coefficient* for that component, the more important that component is to generate the verbatim memorized information since it has a greater impact on the sum in Equation 1.

To effectively compare the different model components, we consider a *relative contribution coefficient* that allows us to compare the importance of the different components with one another. For this reason, we consider the sum of all contributions as a normalizing factor and obtain the coefficient:

$$o_p^l = \frac{o^l}{\sum_{i=1}^L a^i + \sum_{i=1}^L h^i}$$

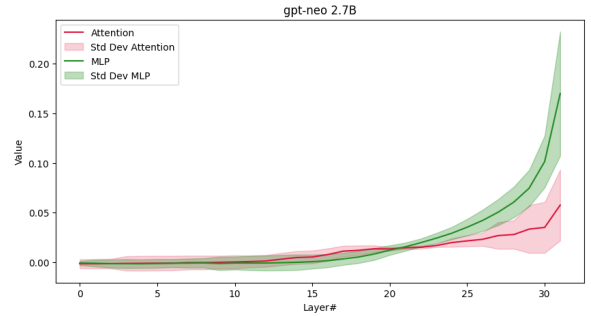


Figure 2: Average contribution of each layer, for Attention and Feed Forward block. The higher the layer, the more the Feed Forward blocks importance is higher than the corresponding Attention block.

In Figure 2 the average *relative contribution coefficient* for each layer for the Feed Forward and Attention block. For the GPT-Neo 2.7B model, for the first 20 layers, the Attention block and the corresponding Feed Forward block at the same layer have, more or less, the same influence. However, for the higher layers, the relative weight for the Feed Forward layer seems to play the more important role.

Those results further testify to the importance of Feed Forward layers in the generation, as also discussed by [\(Geva et al., 2021, 2022\)](#), and we will use this information to inform the edit in PME.

6.3 Association Attacks Prompts

As discussed in Section 3.1, in a Association Attack the model is fed with a prompt that contains an identifier of the person whose information is to

be extracted but that does not exactly match the training data. We adopt this definition as originally proposed by Huang et al. (2022). They propose *zero-shot* attack prompts and defined four slightly different prompts that contain a reference to the identity – the name – of the PII owner and labeled the prompts with letters from *a* to *d*. Then, given a target {name} of an individual, the model is prompted to generate the PII following one of the prompts:

- *a*: the email address of {name} is
- *b*: name: {name}, email:
- *c*: {name} [mailto:
- *d*: —Original Message— From: {name} [mailto:

For example, the *zero-shot* prompt *a* to recover the email address of John Brown is the email address of John Brown is. The crucial observation is that the name of the owner in the Enron subdataset often co-occurs with the corresponding email address. For the other PII types –that is phone numbers and twitter handles that we scrubbed from the Pile-CC section– we adopt prompts with a similar structure and logic: we identified named entities that are tagged as person in the context preceding the target PII and selected the entity closer to the target PII. We will refer to this tagged entity reference as the name in the training example. Then, given the {name} of the identified PII owner, the *zero-shot* prompts for the given {PII type} are:

- *a*: the {PII type} of {name} is
- *b*: name: {name}, {PII type}:
- *c*: {name} at:
- *d*: contact {name} at

While we still expect the Association Attacks to be systematically less effective than Memorization Attacks, we include them to simulate the presence of an almost completely uninformed attacker. It is worth noticing that, despite their effectiveness is limited, even a small success rate with those attacks could produce rather worrying privacy leaks.

6.4 Experimental Design of Extraction of PII Attacks

In our experiments, we define a PII as memorized in terms of the ability of the attack to extract it: this

	Decoding	Configuration	Leak	Tot
Memorization	Greedy	context 50	353	2827
		context 100	476	2932
		context 200	537	2951
	Beam search	context 50	346	2689
		context 100	476	2809
		context 200	515	2863
Association	Greedy	zero-shot a	5	3130
		zero shot b	2	3229
		zero shot c	26	3234
		zero shot d	68	3237
	Beam search	zero shot a	6	3178
		zero shot b	1	3178
		zero shot c	28	3232
		zero shot d	73	3234

Table 5: TDE Memorization and Association Attacks against pre-edit GPT-J 6B . The number of leaked PII **Leak** and the total number of generated PII **Tot** are reported. Given the same prompt **Configuration**, no clear gap can be seen in the two different decoding strategies.

definition – formalized as discoverable memorization (Carlini et al., 2023) – may be influenced by different factors. In fact, the quantified accuracy of the attack depends on the prompt used, the decoding strategy, and the number of times a particular prompt is used (Hayes et al., 2025).

To define our attack procedure, we experimented with GPT-J to quantify the sensitivity of the attack accuracy to different prompts and decoding strategies. In Table 5, the results of TDE attacks against the GPT-J model to extract emails are reported, comparing different prompt lengths for the Memorization Attacks, different zero shot templates in the Associations Attacks, and two different decoding strategies – greedy decoding and beam search – in the pre-edit scenario.

The prompt seems to play a crucial role: the Memorization Attacks are more effective than Association Attacks – in line with previous findings (Huang et al., 2022) – and the length of the prompt plays a strong role (Carlini et al., 2023). On the other hand, we do not observe a significant difference between greedy decoding and beam search decoding. For these reasons, we defer a detailed analysis of the impact of different decoding strategies, as well as the possibility of querying the model with multiple queries for the same PII, as suggested by (Hayes et al., 2025), to future work and focus for the rest of the paper exclusively on discoverable memorization under greedy decoding.

6.5 PME Algorithm

As discussed in Section 2.2, the core component of the PME algorithm, presented in detail in Algorithm 1, is the computation of a contribution score for each layer, that allow PME to reconstruct the privacy preserving value layer by layer in a dense fashion.

The intuition is that each layer should encode a fraction of the final privacy preserving value v^* , and that this fraction should be proportional to the observed contribution of each layer to the generation of the original PII.

This intuition is reinforced by the observation that, if a fixed contribution is defined for each level, PME tends to make the model less robust in terms of post-editing linguistic capacity as the contribution coefficient increases: in Table 6 we demonstrate the role of our contribution coefficient by studying the effect of defining a constant contribution coefficient c .

As discussed in Section 3.3, the post-edit language model should be as similar as possible to the pre-edit model in generations that do not contain private information. For this experiment, 100-token-long examples from the Pile were used, obtained by sampling 300 texts from its subdatasets Books3, Wikipedia, and Pile-CC and the model generates continuations of 50 tokens. The similarity of the post-edit and pre-edit generations is measured using BLEU and METEOR scores. The results in Table 6 demonstrate a decrease in similarity as c increases and further motivate the choice implemented in PME to define a contribution coefficient for each instance and layer.

6.6 Post-edit Association Attacks

In Table 7 results of Association Attacks are presented. Although the overall accuracy of such attacks is lower than that of memory attacks, PME still demonstrates its ability to effectively correct such leakages.

6.7 Post-edit Language Models Abilities

Effective model editing strategies should modify only the patterns of interest, while preserving the LLMs' general abilities and knowledge at the same time. Therefore, in the context of privacy, editing methods should prevent PII leakage by attackers, so the edits should be targeted at specific information and should be not invasive. We compare PME with several editing approaches to understand how

these methods affect the edited LLMs' abilities. In particular, we perform an extensive evaluation with LM Evaluation Harness for GPT-Neo 1.3B (Figure 3), GPT-Neo 2.7B (Figure 4), and GPT-J 6B (Figure 5). In Table 10 we also compare the post-edit generations and the pre-edit ones, measuring their similarity using BLEU and METEOR metrics.

By observing the evaluation results in Figure 3, Figure 4, and Figure 5 and Table 10, we note that DeMem is the baseline performing worse. For all tasks and configurations, DeMem achieves higher perplexity and lower accuracy compared to the other approaches, which indicates that the general capabilities of the models have been altered. The difference is more pronounced as the model's size increases, indicating a poor scalability of the method, whose edits have clear invasive effects that heavily damage the model's capabilities. The similarity of post-edit generation is, across the entire Table 10, always the lower.

Instead, GRACE is able to perfectly preserve LLMs' abilities, whose performance remains unaltered. However, as we discussed also in Section 4.2, this is probably due to the fact that GRACE intervenes only for specific prompts and is not able to generalize, thus avoiding the modification of unrelated behaviors.

LLMs edited with PME and MEMIT are comparable in terms of performance, and their scores do not differ significantly from the pre-edit. Results of GPT-J reported in Figure 5 show that accuracy and perplexity for both PME and MEMIT are nearly identical to the original model for the majority of PII, thus suggesting the efficacy of both methods at performing targeted edits. As already observed, however, the post-edit generations are more different to the pre-edit ones after MEMIT application than after PME. The same pattern can be observed for GPT-Neo-1.3B (Figure 3) and GPT-Neo-2.7B (Figure 4).

6.8 PME is Robust after a large number of edits

As we discussed in Section 4.4, PME is able to largely protect user privacy while maintaining the unaltered LLM capabilities. In Table 8 the results of the TDE Attacks for each of the PII types are detailed: PME is compared with the strongest of the baselines, MEMIT. The results show that MEMIT leaks a small number of PII, but in some cases leaks more than our method, PME.

c	Books3		Wikipedia		Pile-CC	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
0.2	0.899 (0.117)	0.907 (0.113)	0.922 (0.122)	0.934 (0.109)	0.913 (0.117)	0.916 (0.116)
0.5	0.891 (0.117)	0.898 (0.116)	0.91 (0.136)	0.923 (0.117)	0.889 (0.126)	0.889 (0.128)
1	0.866 (0.119)	0.871 (0.123)	0.875 (0.137)	0.889 (0.126)	0.891 (0.115)	0.899 (0.112)
2	0.847 (0.112)	0.851 (0.116)	0.855 (0.14)	0.868 (0.134)	0.856 (0.13)	0.866 (0.121)
3	0.797 (0.096)	0.805 (0.104)	0.82 (0.132)	0.844 (0.121)	0.787 (0.119)	0.8 (0.119)
5	0.662 (0.045)	0.665 (0.048)	0.665 (0.108)	0.661 (0.108)	0.642 (0.082)	0.646 (0.077)
10	0.677 (0.037)	0.661 (0.049)	0.667 (0.095)	0.655 (0.106)	0.653 (0.07)	0.644 (0.076)

Table 6: Reliability of post-edit GPT-J with a constant contribution coefficient c : as c increases, the post-edit generations tend to be less similar to the generations of the pre-edit models, as evidenced by the average BLEU and METEOR scores reported on different subdatasets.

Algorithm 1: The PME Algorithm

Input:

model M autoregressive transformer of L layers, $\mathcal{S} = \{(p, t) \mid \text{s.t. } M(p) = t\}$, dummy PII t^* , estimated keys K_0^l for each layer $l \in [1, \dots, L]$, Feed Forward matrices W_{in}^l and W_{out}^l and activation function f for each layer $l \in [1, \dots, L]$

Output: Post update model M

for $(p, t) \in \mathcal{S}$ **do**

 Record values and contribution to the current output:

 Let $\{x_n^j\}_{j=1}^L$ the output of all layers on input p at last prompt token of index n

for $l \in [1, L - 1]$ **do**

 Compute contribution of layer l : $w_p^l = \frac{x_n^l \cdot x_n^L}{\|x_n^L\|^2}$

 Compute *contribution coefficient*: $w^l = \frac{w_p^l}{\sum_{j=1}^{L-1} w_p^j}$

 Compute target privacy-preserving values x_i^* :

 optimize $\delta^* = \arg \max_{\delta} \mathcal{P}(t^* \mid \sigma((x_n^L + \delta)W_U))$ via Gradient Descent with early stopping

$x^* \leftarrow x_n^L + \delta^*$

for $l \in [1, L - 1]$ **do**

for $(p, t) \in \mathcal{S}$ **do**

 Let a_n^{l-1} the output of the attention block at the previous layer

 Compute keys for the matrix W_{out}^l :

$k^{*l} = m^l = f(W_{in}^l(a_n^{l-1} + x_n^{l-1}))$

 Compute current as: $v_0^{*l} = k^{*l}W_{out}^l$

 Compute new values as: $v^{*l} = w^l x^*$

$K^{*l} \leftarrow [k^{*l}]_{\forall (p,t) \in \mathcal{S}}$

$V_0^{*l} \leftarrow [v_0^{*l}]_{\forall (p,t) \in \mathcal{S}}$

$V^{*l} \leftarrow [v^{*l}]_{\forall (p,t) \in \mathcal{S}}$

$\Delta^l = (V^{*l} - V_0^{*l})K^{*lT}(K_0^l K_0^{lT} + K^{*l} K^{*lT})^{-1}$

$W_{out}^l \leftarrow W_{out}^l + \Delta^l$

Model	Attacks	Pre Edit			PME		MEMIT		GRACE		DeMem	
		Leak	Tot	Acc	Leak	Δ Acc %	Leak	Δ Acc %	Leak	Δ Acc %	Leak	Δ Acc %
GPT Neo 1.3B	email	zero shot a	0	2792	0	0	0	0	0	0	9	
		zero shot b	1	3219	0	0	-100	1	0	1	0	-100
		zero shot c	0	3225	0	0		0	0	0	1	
		zero shot d	16	3232	0.5	0	-100	1	-93.75	16	10	-37.5
	phone	zero shot a	0	65	0	0		0		0	0	
		zero shot b	0	658	0	0		0		0	0	
		zero shot c	0	13	0	0		0		0	0	
		zero shot d	0	997	0	0		0		0	0	
	URL	zero shot a	5	3783	0.1	2	-60	3	-40	5	6	20
		zero shot b	0	1185	0	0		0		0	0	
		zero shot c	2	1803	0.1	1	-50	1	-50	2	1	-50
		zero shot d	3	456	0.7	0	-100	1	-66.67	3	2	-33.33
GPT Neo 2.7B	email	zero shot a	1	1638	0.1	0	-100	0	-100	1	2	100
		zero shot b	1	3230	0	0	-100	0	-100	1	0	-100
		zero shot c	0	3229	0	0		0		0	4	
		zero shot d	40	3238	1.2	0	-100	2	-95	40	16	-60
	phone	zero shot a	0	105	0	0		0		0	0	
		zero shot b	0	89	0	0		0		0	0	
		zero shot c	0	25	0	0		0		0	0	
		zero shot d	0	1905	0	0		0		0	0	
	URL	zero shot a	3	3806	0.1	2	-33.33	6	100	3	2	-33.33
		zero shot b	0	477	0	0		0		0	0	
		zero shot c	1	1104	0.1	0	-100	1	0	1	1	0
		zero shot d	4	495	0.8	0	-100	3	-25	4	3	-25
GPT-J 6B	email	zero shot a	5	3130	0.2	0	-100	4	-20	5	0	-100
		zero shot b	2	3229	0.1	0	-100	3	50	2	1	-50
		zero shot c	26	3234	0.8	0	-100	4	-84.62	26	0	-100
		zero shot d	68	3237	2.1	0	-100	1	-98.53	68	2	-97.06
	phone	zero shot a	0	77	0	0		0		0	0	
		zero shot b	0	92	0	0		0		0	0	
		zero shot c	0	58	0	0		0		0	0	
		zero shot d	0	1618	0	0		0		0	0	
	URL	zero shot a	2	3346	0.1	0	-100	1	-50	2	1	-50
		zero shot b	1	2938	0	0	-100	2	100	1	0	-100
		zero shot c	5	1885	0.3	0	-100	1	-80	5	0	-100
		zero shot d	5	478	1	0	-100	0	-100	5	0	-100

Table 7: TDE Memorization Attacks in pre-edit and post-edit GPT Neo 1.3B, GPT Neo 2.7B, and GPT-J 6B models. In the pre-edit configuration, the number of leaked PII **Leak**, the total number of generated PII **Tot** and the accuracy **Acc %** are reported. For the post-edit attacks, the number of leaked PII **Leak** and the percentage of initially leaked PII that have been successfully removed Δ **Acc %** is reported for each method.

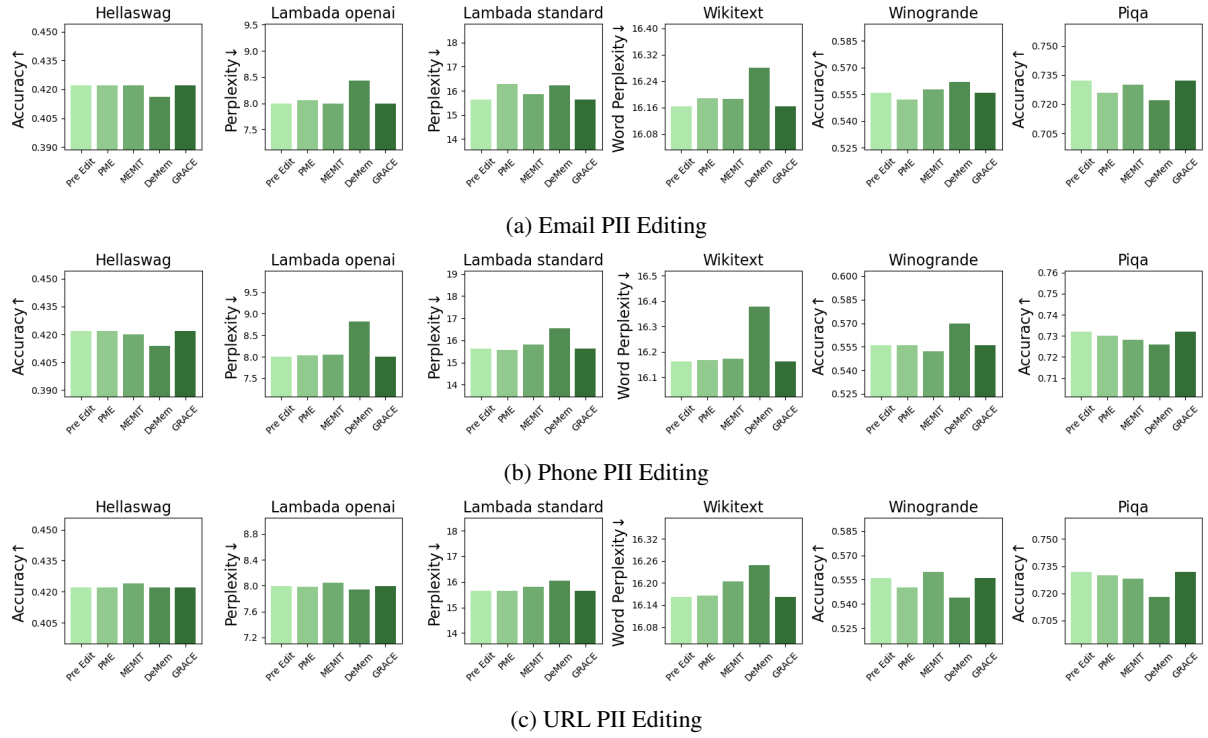


Figure 3: LM Evaluation Harness for GPT-Neo-1.3B Post-Edit

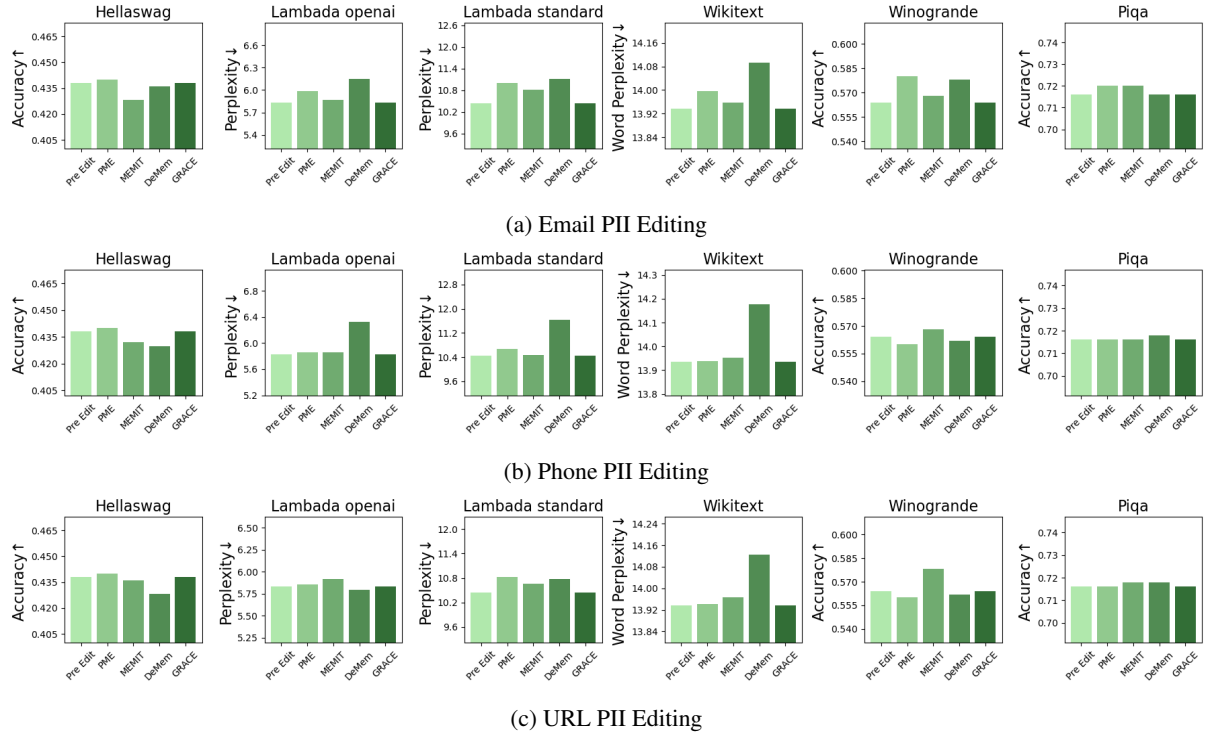
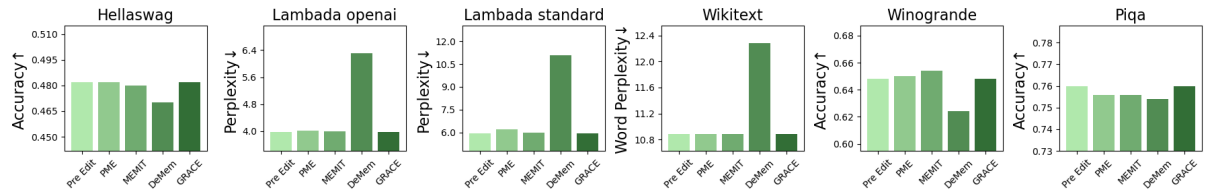
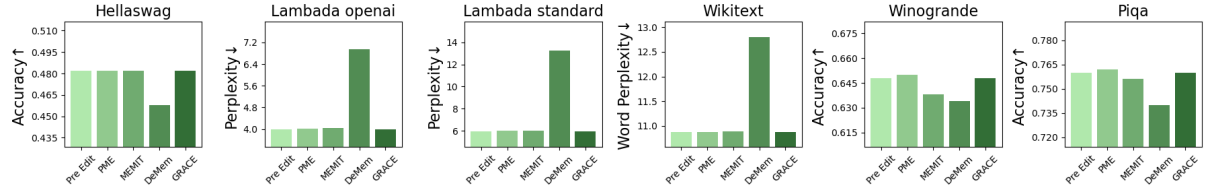


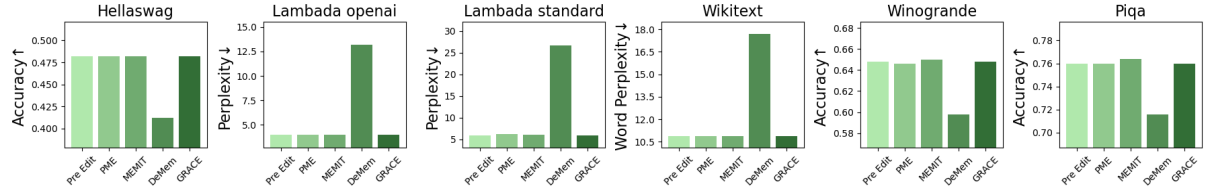
Figure 4: LM Evaluation Harness for GPT-Neo-2.7B Post-Edit



(a) Email PII Editing



(b) Phone PII Editing



(c) URL PII Editing

Figure 5: LM Evaluation Harness for GPT-J 6B Post-Edit

PII Type	Attacks	Pre Edit			PME	MEMIT
		Leak	Tot	Acc	Leak	Leak
email	50	353	2827	0.125	0	0
	100	476	2932	0.162	0	0
	200	537	2951	0.182	0	0
	zero shot a	5	3130	0.002	0	0
	zero shot b	2	3229	0.001	0	0
	zero shot c	26	3234	0.008	0	0
	zero shot d	68	3237	0.021	0	0
phone	50	99	3132	0.032	0	1
	100	125	3166	0.039	1	1
	200	161	3240	0.05	1	0
	zero shot a	0	77	0	0	0
	zero shot b	0	92	0	0	0
	zero shot c	0	58	0	0	0
	zero shot d	0	1618	0	0	0
URLs	50	112	2288	0.049	1	9
	100	148	2327	0.064	1	5
	200	168	2333	0.072	1	4
	zero shot a	2	3346	0.001	0	1
	zero shot b	1	2938	0	0	2
	zero shot c	5	1885	0.003	0	0
	zero shot d	5	478	0.01	0	0

Table 8: TDE Attacks in pre-edit and post-edit for the GPT-J 6B model after the edit of all the PII. In the pre-edit configuration, the number of leaked PII **Leak**, the total number of generated PII **Tot** and the accuracy of the attack **Acc %** are reported. For the post-edit attacks, the number of leaked PII **Leak** is reported for PME and MEMIT

Context		50			100			200		
		email	URL	phone	email	URL	phone	email	URL	phone
Pre-edit	correct prediction	353	112	99	476	148	125	537	168	161
	PII predicted	2827	2288	3132	2932	2327	3166	2951	2333	3240
PME	correct prediction	0	0	0	0	0	0	0	0	0
	PII predicted	57	7	10	44	3	7	39	9	8
MEMIT	correct prediction	0	4	0	0	1	0	0	1	0
	PII predicted	120	186	116	65	205	121	57	204	115

Table 9: New PII predicted after the edit procedure of the GPT-J model via Memorization Attacks, detail for each PII type.

Model	PII	Edit	Books3		Wikipedia		Pile-CC	
			BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
GPT Neo 1.3B	email	PME	0.925 (0.103)	0.93 (0.102)	0.941 (0.097)	0.946 (0.094)	0.897 (0.119)	0.907 (0.111)
		MEMIT	0.92 (0.102)	0.924 (0.103)	0.904 (0.135)	0.916 (0.118)	0.896 (0.114)	0.905 (0.108)
		GRACE	0.989 (0.057)	0.989 (0.056)	1.0 (0.0)	1.0 (0.0)	0.997 (0.033)	0.997 (0.032)
		DeMem	0.864 (0.117)	0.87 (0.121)	0.875 (0.123)	0.892 (0.113)	0.828 (0.122)	0.846 (0.118)
	phone	PME	0.95 (0.096)	0.953 (0.095)	0.966 (0.084)	0.965 (0.09)	0.927 (0.117)	0.936 (0.106)
		MEMIT	0.881 (0.12)	0.89 (0.12)	0.92 (0.124)	0.93 (0.107)	0.895 (0.122)	0.902 (0.117)
		GRACE	0.989 (0.057)	0.989 (0.056)	1.0 (0.0)	1.0 (0.0)	0.997 (0.033)	0.997 (0.032)
		DeMem	0.813 (0.106)	0.824 (0.111)	0.835 (0.132)	0.85 (0.128)	0.796 (0.126)	0.813 (0.121)
	URL	PME	0.957 (0.089)	0.959 (0.089)	0.975 (0.068)	0.977 (0.066)	0.938 (0.113)	0.943 (0.106)
		MEMIT	0.882 (0.116)	0.891 (0.117)	0.887 (0.136)	0.899 (0.123)	0.862 (0.136)	0.864 (0.131)
		GRACE	0.989 (0.057)	0.989 (0.056)	1.0 (0.0)	1.0 (0.0)	0.997 (0.033)	0.997 (0.032)
		DeMem	0.841 (0.114)	0.853 (0.115)	0.866 (0.132)	0.882 (0.126)	0.82 (0.129)	0.835 (0.123)
GPT Neo 2.7B	email	PME	0.906 (0.112)	0.912 (0.113)	0.922 (0.111)	0.931 (0.104)	0.87 (0.123)	0.879 (0.123)
		MEMIT	0.895 (0.123)	0.897 (0.127)	0.914 (0.101)	0.925 (0.095)	0.885 (0.121)	0.882 (0.128)
		GRACE	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
		DeMem	0.817 (0.109)	0.822 (0.115)	0.83 (0.12)	0.847 (0.121)	0.81 (0.132)	0.82 (0.128)
	phone	PME	0.942 (0.093)	0.944 (0.094)	0.946 (0.102)	0.957 (0.076)	0.905 (0.127)	0.908 (0.123)
		MEMIT	0.905 (0.115)	0.91 (0.114)	0.925 (0.11)	0.937 (0.095)	0.872 (0.128)	0.878 (0.125)
		GRACE	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
		DeMem	0.796 (0.096)	0.804 (0.105)	0.82 (0.119)	0.835 (0.121)	0.779 (0.124)	0.783 (0.124)
	URL	PME	0.928 (0.101)	0.931 (0.103)	0.912 (0.123)	0.931 (0.095)	0.872 (0.134)	0.879 (0.132)
		MEMIT	0.89 (0.116)	0.894 (0.117)	0.907 (0.11)	0.922 (0.094)	0.833 (0.116)	0.84 (0.12)
		GRACE	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
		DeMem	0.803 (0.101)	0.811 (0.109)	0.837 (0.121)	0.862 (0.119)	0.788 (0.11)	0.797 (0.113)
GPT-J 6B	email	PME	0.945 (0.093)	0.947 (0.096)	0.954 (0.094)	0.959 (0.09)	0.946 (0.096)	0.95 (0.095)
		MEMIT	0.902 (0.108)	0.91 (0.107)	0.906 (0.124)	0.916 (0.117)	0.912 (0.118)	0.914 (0.112)
		GRACE	0.988 (0.06)	0.988 (0.059)	1.0 (0.0)	1.0 (0.0)	0.997 (0.032)	0.997 (0.029)
		DeMem	0.742 (0.06)	0.746 (0.077)	0.749 (0.118)	0.763 (0.121)	0.726 (0.089)	0.732 (0.096)
	phone	PME	0.953 (0.092)	0.955 (0.09)	0.962 (0.082)	0.966 (0.081)	0.951 (0.096)	0.956 (0.088)
		MEMIT	0.858 (0.116)	0.864 (0.119)	0.869 (0.136)	0.883 (0.126)	0.849 (0.121)	0.859 (0.117)
		GRACE	0.988 (0.06)	0.988 (0.059)	1.0 (0.0)	1.0 (0.0)	0.997 (0.032)	0.997 (0.029)
		DeMem	0.725 (0.041)	0.732 (0.059)	0.73 (0.112)	0.747 (0.11)	0.706 (0.094)	0.722 (0.091)
	URL	PME	0.935 (0.093)	0.939 (0.093)	0.904 (0.123)	0.917 (0.111)	0.898 (0.125)	0.907 (0.119)
		MEMIT	0.853 (0.112)	0.856 (0.115)	0.878 (0.127)	0.895 (0.114)	0.833 (0.122)	0.84 (0.124)
		GRACE	0.988 (0.06)	0.988 (0.059)	1.0 (0.0)	1.0 (0.0)	0.997 (0.032)	0.997 (0.029)
		DeMem	0.723 (0.055)	0.735 (0.071)	0.734 (0.117)	0.757 (0.123)	0.694 (0.089)	0.712 (0.091)

Table 10: Reliability of post-edit LLMs for all the considered baselines.