Article

Not peer-reviewed version

Enhancing Data Privacy in Large Language Models through Private Association Editing

Davide Venditti [*,†], Elena Sofia Ruzzetti [†], Giancarlo A. Xompero, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, Fabio Massimo Zanzotto

Posted Date: 4 March 2026

doi: 10.20944/preprints202603.0339.v1

Keywords: anonymity and privacy; model editing; locate-then-edit paradigm; training data extraction attacks; artificial intelligence security

 doi:10.20944/preprints202603.0339.v1

Article

Enhancing Data Privacy in Large Language Models through Private Association Editing

Davide Venditti [1,*,†], Elena Sofia Ruzzetti [1,†], Giancarlo A. Xompero [1,2,†], Cristina Giannone [2], Andrea Favalli [2], Raniero Romagnoli [2] and Fabio Massimo Zanzotto [1,2]

[1] Human Centric ART, University of Rome Tor Vergata, Italy
[2] Almawave S.p.A., Via di Casal Boccone, 188-190 00137, Rome, Italy
* Correspondence: davide.venditti@uniroma2.it
† These authors contributed equally to this work.

Abstract

Large language models (LLMs) require a significant redesign in solutions to preserve privacy in data-intensive applications due to their text-generation capabilities. Indeed, LLMs tend to memorize and emit private information when maliciously prompted. In this paper, we introduce Private Association Editing (PAE) as a novel defense approach for private data leakage. PAE is designed to effectively remove Personally Identifiable Information (PII) without retraining the model. Experimental results demonstrate the effectiveness of PAE with respect to alternative baseline methods. We believe PAE will serve as a critical tool in the ongoing effort to protect data privacy in LLMs, encouraging the development of safer models for real-world applications.

Keywords: anonymity and privacy; model editing; locate-then-edit paradigm; training data extraction attacks; artificial intelligence security

1. Introduction

Preserving privacy a major challenge for designers of data-intensive applications, and thus, it is a well-studied topic with well-established solutions. After meeting the applications' requirements, their focus immediately shifts to safeguarding users' privacy. Indeed, the association between Personally Identifiable Information (PII) and related data must be used following permissions granted by users and, possibly, enforced by law[1].

Privacy is generally preserved by controlling access to the data repository and securing the communication channels, utilizing privacy-by-design [1–4] or cryptography [5–8].

Large language models (LLMs) require a significant shift in the solutions to preserve privacy due to their text-generation capabilities. Protecting training data and learned models in secure servers is not sufficient, as may be for machine learning classifiers.

In fact, LLMs can unintentionally reveal associations between PII and sensitive information when prompted in specific ways [9,10]. Their strength in abstraction and memorization [11–13], paired with training on extensive web-scraped data as Common Crawl [14], increases the likelihood of disclosing private information. Thus, designing privacy-preserving LLMs that can appropriately manage PII and sensitive data is a critical and compelling challenge that must be addressed [15]. In this paper, we propose a novel model to preserve privacy in LLMs: *Private Association Editing* (PAE), a "one model, k edits" strategy to remove memorized private information adjusting parameters of LLMs without re-training (see Figure 1). PAE is a model-editing privacy-preserving strategy based on the idea of *breaking the association* between personal information and the identity of the person to whom it belongs by replacing the original information with masked – but semantically equivalent – information.

[1] EU with the General Data Protection Regulation (GDPR), the US with the Privacy Act, and China with the Personal Information Protection Law and Data Security Law

 doi:10.20944/preprints202603.0339.v1

Inspired by recent model editing techniques [16,17], PAE proposes two main innovations: the PAE cards and the PAE Regularization strategy. Experiments with GPT-J [18] and GPT-Neo [19] show that PAE outperforms alternative baseline methods in reducing privacy leaks without degrading the capabilities of LLMs to generate texts.

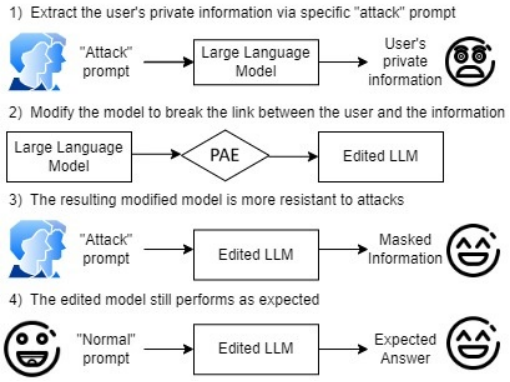

Figure 1. Preserving privacy for LLMs by using Private Association Editing

The major contributions of the paper are:

- An innovative strategy to reduce privacy leak risks in LLMs: the PAE method that extends beyond factual editing approaches;
- Two important components of the PAE Method: PAE Cards and PAE regularization;
- The experimental analysis showing that PAE is an effective method to reduce privacy leaks and outperforms existing baseline methods.

2. Background and Related Work

2.1. Privacy Issues with LLMs

Large Language Models (LLMs) are prone to emitting private information that has been included in their training material. Indeed, attacking LLMs to extract memorized private information is possible by using black-box access to language models. Training Data Extraction (TDE) is a technique to extract this private information [20]. It consists of querying the target model to force it to produce its own training data and –as a result– personal information inadvertently included in the training data like Twitter handles and email addresses [20]. These attacks are more effective if the private information to extract is preceded by the original training sequence in which it appeared [10]. Huang et al. [21] demonstrated that conditioning a model with a prompt that is part of the training data can result in the leakage of personally identifiable information (PII), such as email addresses. Nasr et al. [22] revealed that Carlini et al. [20] method is even more effective than previously expected: by querying open-source models like GPT-Neo [23] and Pythia [24], they confirmed the success of the attack procedure using the training data solely for verification purposes. Since these attacks require only black-box access to the model, closed models like GPT-3.5 and GPT-4 can be successfully attacked [25].

2.2. Strategies to Protect Privacy in LLMs

As personal information leakage from LLMs is a concrete possibility, different strategies have been explored to avoid a model generating potentially harmful content.

During the training phase, formal guarantees of privacy preservation are provided by Differential Privacy (DP) methods. These methods ensure that the model's outputs are statistically indistinguishable, with high probability, when any single individual's data in the training set is modified, added, or removed. This property limits the extent to which information about individual training examples can be inferred from the trained model. While some work applied DP to Transformer-based language models during the pre-training phase [26,27], DP training is computationally expensive, and its appli-

 doi:10.20944/preprints202603.0339.v1

cation remains challenging: the majority of already trained and distributed models, not implementing DP, remain vulnerable.

Ideally, private data could not be included in the training phase at all: a number of works focus on modifying input text to obfuscate sensitive attributes before training. Solutions range from classical anonymization with Named Entity Recognition techniques [28], to privacy-aware text rewriting leveraging modern, Transformer-based, language models [29]. However, such data-level interventions can no longer be applied after the model training is finished, and models already trained without implementing data anonymization techniques remain vulnerable. Furthermore, the scalability of such techniques on large datasets, such as those used during pre-training of LLMs, remains unexplored [30].

In case PII were inadvertently included in the pre-training set, the most intuitive approach would be to retrain the model from scratch after performing a data cleaning process to remove the identified PII. Retraining from scratch after removing such information, however, is computationally expensive and unfeasible for large models. Moreover, recent work also discusses that removing some private information during training may also cause new leakages [31].

To avoid the expensive remove-and-retrain procedure, a possible solution is offered by unlearning algorithms. Yao et al. [32] propose an unlearning mechanism that requires only negative samples – i.e., examples in which the model generates harmful content – to stop the generation of undesirable outputs. However, the approach proposed by Yao et al. [32], as the majority of machine unlearning approaches [33], requires the definition of a retain set that contains samples used to preserve the utility of the model. Our aim is to modify only a batch of information without further training or additional data. Among the unlearning algorithms that do not require additional data, DeMem [34] is based on reinforcement learning, to make the model rewrite text that is privacy-preserving (we will discuss this method as a baseline of our proposed approach in Section 4.4).

2.3. Knowledge Editing in Transformers

Model editing is a possible solution as opposed to an expensive remove-and-retrain strategy and has been extensively used to update factual knowledge in LLMs [35].

A possible solution is to train a hyper network to compute model updates that encode new factual knowledge. Cao et al. [36] edit factual knowledge within language models, ensuring consistency across various formulations of facts. Similarly, Yao et al. [37] introduced MEND algorithm and demonstrated its ability to effectively edit large-scale models' behaviors, modifying fine-tuning gradients into a specific parameter update to edit a certain factual proposition. We will discuss MEND as one of our baselines in Section 4.4.

Building on the idea that the linear layers in the Transformer architecture can be interpreted as key-value memories that store information [38], ROME [16] and MEMIT [17] demonstrate the ability to edit factual knowledge. Since these methods can modify factual information memorized in LLMs, our goal is to exploit them to erase private information inadvertently ingested during training.

2.4. Model Editing for Privacy Protection

In some cases, neurons' activations are zeroed out to avoid the generation of private information, like in DEPN [39]. However, it has also been discussed that single neurons are more likely to encode different features, rather than a single one [40,41], so zeroing their value could lead to hindering their performance on unrelated tasks. Moreover, while this approach can lead to disabling the generation of the PII, controlling the language model to generate a different target value is not directly possible using this strategy. Wu et al. [42] present a novel framework called Augmented Privacy Neuron Editing via Activation Patching (APNEAP), designed to safeguard data privacy while maintaining high performance. However, this study is not directly relevant to our research, as they rely on activation patching, whereas we focus on directly modifying the model's weights. In fact, these approaches foresee the use of external modules that are clearly separated from the model's weights, thus they can be removed easily to get access to the original parameters.

 doi:10.20944/preprints202603.0339.v1

Private Memorization Editing [43] also defines a model editing technique to directly remove private information memorized in model weights. However, unlike their methodology, PAE does not rely on direct knowledge of models' pre-training data during the *editing* phase. In fact, as we will discuss in Section 3.2, our PAE cards define prompts that can be constructed without any reliance on the pre-training material but the name of the data owner.

Patil et al. [44] investigated model editing techniques to modify the information memorized in LLMs. They experimented with ROME and MEMIT and concluded that information could not be erased. In particular, they applied TDE attacks against the GPT-J [18] model and demonstrated that in black-box access–performing attacks that also include paraphrases of the original prompt– model editing cannot erase factual information memorized in GPT-J. Our setting is different: in fact, Patil et al. [44] investigated the effectiveness of model editing only on factual information from sentences derived from Wikipedia and not directly present in the training data – the Pile [45]. By definition, the model under attack does not *verbatim* memorize information that is not in training data: since the examples used by Patil et al. [44] are derived from Wikipedia and not included in the Pile, while the factual information they contain is memorized, they cannot be verbatim memorized. In our experiments, we directly study the effectiveness of model editing in deleting private information that is verbatim memorized rather than factual information.

3. Materials & Methods

In this Section, we discuss our framework to attack LLMs and our proposed technique to prevent data leakage. Large Language Models (LLMs) have a tendency to emit private information memorized from their training data when fed with malicious prompts. In Training Data Extraction (TDE) attacks, if a model is prompted with a prefix encountered during training, it often completes it with the rest of the training sequence by producing verbatim private information [10,21].

In this scenario, we propose a model to remove memorized Personally Identifiable Information (PII) from LLMs and thus reduce possible privacy leaks. Our procedure is extremely more versatile than remove-and-retrain and can be used in small batches of edits of an LLM. Since the editing phase may cause negative effects on the general language models ability, as part of our pipeline we validate that such capabilities are preserved [46–48]. Similarly to Ruzzetti et al. [43], it consists of three steps (see Figure 1):

- detecting the presence of memorized PII in *pre-edit* LLMs performing black box TDE attacks (Section 3.1);
- *Private Association Editing* (PAE) to remove PII by editing parameters of LLMs obtaining *post-edit* LLMs (Section 3.2)
- a final consistency check of *post-edit* LLMs to assess that LLMs are not corrupted after PAE and behave similarly to *pre-edit* LLMs (Section 3.3)

3.1. Training Data Extraction Attacks to Recover Sensitive Information

To detect the presence of memorized Personally Identifiable Information (PII) in LLMs, we follow the attack pipeline and attack prompts defined by Huang et al. [21]. They defined two kinds of attacks depending on how information is stored and retrieved: (1) a model *memorizes* personal information if there exists a prompt from the training data that leads the model to generate that information; (2) in contrast, a model *associates* an individual with its personal information if there exists a prompt not seen during training but containing a reference to an individual that leads to the generation of PII. Huang et al. [21] already demonstrated that memorization is more common in LLM than association, showing that a model from the GPT-Neo family [19] can predict emails more accurately when conditioned with prompts from the training data rather than with unseen prompts.

We then analyze two attacking schemes: the Memorization attacks and the Association attacks.

In a **Memorization attack**, a model is fed with a prompt extracted from its pretraining data. This prompt is the *context* that precedes the private PII in the training data. For example, a *context* prompt

 doi:10.20944/preprints202603.0339.v1

attack to recover the email address of `John Brown` would look like: `"All the winter months might settle 2.25. As such, the best thing to be short is jan. ---Original Message--- From: John, Brown"`. The *context* prompt is characterized by a length defined in terms of tokens. In this attack, following Huang et al. [21], we simulate that the attacker has more or less knowledge about the training data by conditioning the generation of the model to *context* prompts of different lengths in terms of tokens. The larger the context, the more effective the attacks turn out to be.

In the **Association Attack**, the model is instead 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. In particular, Huang et al. [21] defined four *zero-shot* attack prompts all containing a reference to the identity – the name – of the PII owner. Those attacks are identified as *zero-shot*, in analogy with the idea that the model is asked to associate the PII owner name with the PII itself without any in-context demonstration. In the association attacks, different prompts are tested and labeled with letters from *a* to *d*. 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"`. A key observation is that, in the Enron sub-dataset, the owner's name frequently appears near the corresponding email address.

For other PII types, we define similar attack prompts to the one proposed by Huang et al. [21]: 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`

For these PII types, to identify the `{name}` of the owner, we tagged the named entities in the context where the PII appears in the same context that it is used in the memorization attacks as well (additional details in Section 4). Among the entities tagged as "person" in this context, we selected the entity closer to the target PII.

The attack succeeds if, during the generation of the subsequent tokens, the model generates the target's private information, that is, the correct email address. In both Memorization and Association attacks, the adversary with black-box access forces the model to generate some PII regarding a person. The analyzed framework encompasses a malicious attacker – or any individual aiming to detect unauthorized use of their data – who has assumptions about the original text that was used during training (in the Memorization Attacks) or who has no prior clues about the original data that contained the private information but who has some other knowledge about the identity of the individual whose sensitive information they wish to extract (in the Association Attacks).

3.2. Private Association Editing as Efficient Defense against Privacy Attacks

To protect data owners from privacy attacks performed on LLMs, we propose *Private Association Editing* (PAE), an editing technique aiming at disrupting *private associations*, i.e., associations between an individual and a PII included in the dataset used to train the LLM. The technique proposed here is efficient since it allows the anonymization of private information directly into the model parameters, without retraining.

In this work, we define a *private association* as an association between the name of an individual and a PII that should not be revealed when interrogating the LLM. We stem from the definition of association between a data owner and its PII as defined in Association attacks (see Section 3.1): the model is able to generate a PII when prompted with some information regarding the data owner, like

 doi:10.20944/preprints202603.0339.v1

its name. Our proposed technique, PAE, aims to leverage the association capabilities of a model to *protect* the privacy of data owners, breaking such association.

The PAE cards (depicted in Figure 2) – for example `"The email address of John Smith is` john.smith@company.com – describe this association between a person's name and their PII. The PAE cards are the first component of our defense strategy.

Then, we propose the PAE Update Strategy on Model's Weights to mask the private information of individuals that has been inadvertently inserted into the training data. PAE Update Strategy allows for the substitution of the PII with a semantically equivalent but anonymous value.

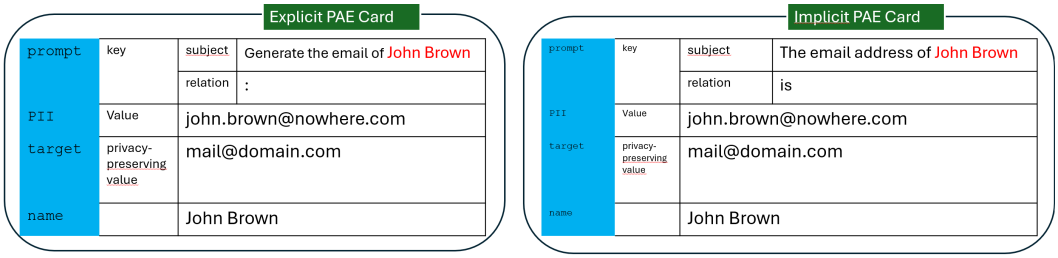

Figure 2. Private Association Editing cards with two prototypes (Implicit and Explicit versions on email addresses)

Parameters of Transformers Store PII

Recently, the phenomenon of memorization in LLMs has been identified as a relevant topic in interpretability research [22,49,50]. This behavior refers to the model retrieving specific information encountered during training when a particular textual pattern is detected in the input prompt. Since such information may include undetected PII, this mechanism could potentially be used to recover sensitive data learned during training [10,20–22].

Interpretability literature show that factual knowledge in transformers is stored in the Feed Forward components (FFN) in the form of key-value pairs (k, v) called *memories* [16,17,38,51], where linear transformations of the prompt representation – a key k – retrieve the associated relevant information – the value v – used by the LLM to generate the next sequence. Building on this assumption, Meng et al. [16,17] proposed a method to modify the information stored in FFN matrices using a Constrained Least Squares optimization approach, which allows to update stored facts. For further details, see Appendix A.1.

PAE Cards to Edit Private Associations

Based on the evidence that the projection matrix of FF Layers store factual knowledge, we assume that associations between individuals and their PII are represented in Transformer-based LM as key-value mappings as well.

Our procedure to edit private associations is based on the use of PAE cards, intuitive data structures reporting the information needed to edit a specific association, typically composed of the following key elements: a `prompt`, `the Personally Identifiable Information (PII)`, a `target` (the desired output), and a `subject` (the individual associated with the information). The PAE Cards are summarized in Figure 2. The `prompt` is a template that is compiled with the `subject` to perform the edit.

We hypothesize that the `subject` is associated with the original `PII` in the projection matrix of FF layers of the Transformer-based LM, and we want to obfuscate the `PII` with our privacy-preserving `target`. Specifically, this association is encoded in the projection matrix as a key-value pair (k, v) with k being the vector representing the `subject` in the compiled `prompt`, and v the vector representing its associated `PII`. Hence, we can also manipulate these associations to prevent the leakage of sensitive data by inserting new updated association (k, v^*), where v^* is the privacy-preserving `target` that is semantically similar to the original PII.

We propose Implicit and Explicit PAE Cards: we stem from prior literature that establishes a distinction between explicit prompts – that convey clear, direct instructions, typically resulting

 doi:10.20944/preprints202603.0339.v1

in specific, fact-based answers – and implicit prompts – that employ subtle, indirect formulations or contextual cues, leading to broader and more flexible responses [52,53]. The Explicit PAE Card is defined as a direct command that clearly and unambiguously conveys the intended instruction, framing the edit as a specific, immediate generation action. The explicit prompt used is "`Generate the {PII type} of {name}:`". Conversely, the Implicit PAE Card employs an indirect formulation that provides indirect cues for the completion: we frame the edit as a text generation task, given the person's name. Our implicit prompt is "`The {PII type} address of {name} is`".

PAE Update Strategy on Model's Weights

PAE updates the Feed-Forward (FF) modules of target LLMs given the PAE Cards. In fact, studies suggest that the FF modules store information in the form of *key-value* memories [16,17,38], as described above and in Appendix A.1. Thus, PAE edits matrices W_0^l, the last projection matrices in the FF module, to change memorized information: $\widehat{W}_0^l = W_0^l + \Delta$, where l is the index layer, omitted when not necessary.

The update matrix Δ should break the association between a *key* encoding a `prompt` and its corresponding *value* encoding the `PII` (see Figure 2). To do so, PAE aims to substitute the current *value* with a new, *privacy-preserving value* anonymous `target`, that is semantically equivalent to the `PII` but does not violate any user privacy.

To determine Δ, the target matrix W_0 should be written as the mapping between a set of keys K_0 and values V_0 learned during the pretraining phase $W_0K_0 = V_0$ [16,17]. Hence, Δ matrix for PAE is defined as a function of the *keys* $K \subset K_0$, the private *values* $V \subset V_0$, and the new *privacy-preserving values* V^*.

For PAE update strategy, we can frame the problem of finding the optimal update Δ to encode the privacy-preserving values V^* imposing that the optimal post-edit matrix $\widehat{W}^*$ – defined as $\widehat{W}^* = W_0 + \Delta$ – should be minimizing the following equation [17]:

$$\sum_{k \in K_0, v \in V_0} ||\widehat{W}k - v||^2 + \sum_{k \in K, v^* \in V^*} ||\widehat{W}k - v^*||^2 \tag{1}$$

and keep the values $V^* = \widehat{W}^*K$ similar to $V = W_0K$. Our update matrix Δ is then computed as:

$$\Delta = \Lambda \otimes (V^* - W_0K)K^T(C_0 + KK^T)^{-1} \tag{2}$$

where Λ is a diagonal matrix defined as a function of the norm of V and V^*, and $\otimes$ is the Hadamard product. This equation is obtained as follows. As discussed by Meng et al. [17], the solution of Equation 1 can be written in a closed form as:

$$\Delta' = (V^* - W_0K)K^T(C_0 + KK^T)^{-1} \tag{3}$$

However, the relative weight of the two members in the sum to compute the post-edit matrix $\widehat{W}^* = W_0 + \Delta$ plays a crucial role: the larger the weight of one of the two components, the larger the similarity of the post-edit matrix $\widehat{W}^*$ with respect to the update Δ or to the pre-edit matrix W_0. Intuitively, if the update is computed as $\widehat{W}^* = W_0 + \Delta'$, scaling Δ' by a constant multiplier λ, the post-edit model will be less consistent than the pre-modification model as λ increases (since the relative weight of W_0 decreases). This causes the post-edit model to diverge rapidly with respect to the respective pre-edit model. Conversely, the post-edit model will be more consistent with respect to the pre-edit one when λ is closer to 0. In Figure 3, we observe that the similarity between the generations of the pre-edit model and the generations of the post-edit ones rapidly decreases as λ increases: we measure the similarity by computing the average BLEU score of the generations, having as reference the pre-edit one. As we will also discuss in Section 3.3, very dissimilar generations are a symptom of a decreased model utility: PAE aims to obtain an equivalent model to the pre-edit one, but capable of preserving users' privacy. On the other hand, for values of λ closer to 0, the post-edit model utility will

 doi:10.20944/preprints202603.0339.v1

be increased (see $\lambda = 0.5$ and $\lambda = 0.2$ in Figure 3). We stem from these observations to introduce a scaling factor to take into account this phenomenon.

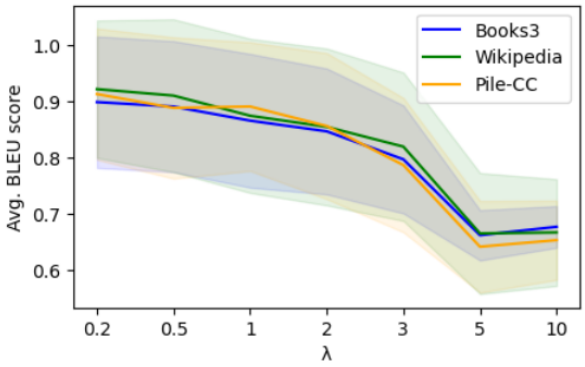

Figure 3. The post-edit model is increasingly different from the pre-edit model as λ increases: this is an indication of a diminished utility of the model.

Then, in PAE, we introduce a mechanism for adjusting Δ' in function of the relative weight of the new values V^* and the old values V, thus obtaining $\Delta = \Lambda \otimes \Delta'$. To design Λ, we analyze the $V^* - W_0K$ term of Equation 2: since by definition $W_0K = V$, the term can be rewritten as $V^* - V$. Hence, the i-th row of the matrix $V^* - V$ quantifies how different the *privacy-preserving* value v_i^* is from the corresponding v_i. We argue that the direction of this difference is important but the norm of the residuals should be comparable to the norm of the values before the update. Then, after the update, the new values should be encoded in a similar way to what was done before the update. Hence, we define the diagonal entry of Λ term as:

$$\Lambda_{i,i} = \frac{||v_i||}{||v_i^* - v_i|| + ||v_i||} \tag{4}$$

to perform both the normalization and scale proportionally to $||V||$. The update rule in Equation 2 allows to preserve users' privacy while maintaining the LLM utility (a detailed algorithm is in Appendix A.2).

When using PAE, we adopt a strategy that we call "one model, k edits": we are interested in subjecting the model to k edits at the time to comply to the real word scenario in which – instead of performing single edits separately and recreating the model based on the post-edit weights obtained from the last edit every time – k different requests are addressed against a single model. As described in Section 4, the k in PAE is not predetermined.

By masking and anonymizing the email address, we make it more challenging for attackers to elicit specific private data from the model in response to particular prompts. This methodology effectively reduces the risk of sensitive information being inadvertently disclosed by the model.

3.3. Evaluating Post-Edit Language Modeling Performance

The final step of the procedure is to investigate whether the LLM maintains its behavior in text generation after the editing process. In fact, Model Editing techniques, in general, and PAE, in particular, may perturb the language model capabilities due to the intervention on the model parameters. The LLM assessment procedure we describe in this Section aims to verify that the privacy-preserving language model is not a worse model than the original one. Since the models under investigation are foundational models, we focus on their language modeling capabilities rather than on an evaluation based on task performance. If after the update the language model performs similarly to the pre-edit one, then also the performance on tasks will be similar.

We first introduce a metric for language model ability that can be used to assess the *post-edit reliability*. The edit should cause no harm to the utility of the LM: to quantify this aspect, we adopt the LAMBADA [54] benchmark. LAMBADA measures the language modeling ability of a model,

 doi:10.20944/preprints202603.0339.v1

calculating the accuracy the model has when asked to generate a missing target word from a passage. In the test split of the dataset, the missing word is always the last in the passage. We will use the LAMBADA test set as the first indicator of the reliability of the edit.

However, we argue that the post-edit should not only demonstrate similar task performance but generate *texts* as similar as possible to the pre-edit one: ideally, we would like to have the post-edit model indistinguishable from the pre-edit one. The evaluation procedure we define is hence based on an automatic comparison between *pre-edit* version LLM and the *post-edit* version LLM. The idea is to collect generations for a given set of prompts for *pre-edit* LLM and *post-edit* LLM. Then, these generations are compared with string-based similarity metrics, in particular BLEU and METEOR metrics. With these measures, we can automatically assess if *pre-edit* LLM and *post-edit* LLM behave similarly. In Appendix A.3, we show that our method correlates with human judgments: for systems that achieve a high similarity in terms of BLEU or METEOR scores, annotators can only guess randomly whether the test examined is generated by the pre-edit or post-edit model.

4. Experimental Setup

In this Section, we define and motivate the setting of our experiments to evaluate the reliability and effectiveness of our approach. We first present the LLMs and related datasets we consider for the analysis (Section 4.1), then we discuss the application of PAE (Section 4.2), we provide details about the setup for evaluating the reliability of LLMs in Post-Edit (Section 4.3), and finally we introduce the baseline privacy-protection methods we consider to compare PAE against (Section 4.4). For our experiments, PAE and the baselines are largely implemented with an open-source package called *EasyEdit*[2]. Further details about the environment used for the experiments are reported in Appendix A.4.

4.1. Analyzed LLMs and TDE

In our experiments, we test the GPT-J model [18] that is designed to generate human-like text continuations from prompts: it is a large model, with 6 billion parameters, trained on the open dataset Pile [45]. The Pile is a large-scale text corpus that aggregates various sources, including books, articles, websites, and scientific papers. To test how scale influences our proposed method, we also test two smaller models belonging to the GPT-Neo family [19] – 1.3 billion and 2.7 billion parameters each: those models are also trained on Pile.

The choice of the models and dataset is crucial since, to effectively measure the performance of the attack, it is necessary to observe the training data [20,22]. However, this requirement is for evaluation purposes and does not limit the applicability of PAE.

One of the constituent sub-datasets within The Pile is the Enron Emails [55] corpus. This dataset contains text from approximately 150 users. It includes a total of about 0.5 million email messages. Its inclusion in the Pile mimics the inadvertent insertion into the training data of private information, in particular of PII-like email addresses: the Enron dataset represents a natural starting point to test GPT-J and GPT-Neo memorization of PII. We also scraped[3] another subsection of Pile, Common Crawl (CC), to identify other PII potentially memorized by the target LLMs: we extracted from Pile-CC phone numbers and Twitter handles from this subset. Our dataset is hence composed of 3333 email addresses, 1635 phone numbers, and 931 Twitter handles. Since most of the data in Pile is in English, our evaluation is limited exclusively to the English language.

We perform TDE as discussed in Section 3.1 to extract the email addresses, phone numbers, and Twitter handles. We focus on greedy decoding since a preliminary study suggests no difference in attack accuracy between greedy and beam search decoding (see Appendix A.5). For Memorization Attacks, we focus on *context* prompts of 200, 100, and 50 tokens. For Association Attacks, we leverage the attack prompts defined in Huang et al. [21] for emails and similarly designed prompts for the other

[2] https://github.com/zjunlp/EasyEdit
[3] Original data from from this dataset, and verified by us scraping Pile-CC

 doi:10.20944/preprints202603.0339.v1

PII types, as described in Section 3.1. We will report the results against each of these prompts with letters from *a* to *d*.

4.2. Application of PAE

PAE edits aim to cover the real-world scenario in which multiple privacy leakages are to be updated in a single edit, following a "one model, k edits" philosophy. The are two distinct ways to apply model editing: *batch* editing that involves editing k elements in an LLM simultaneously; *sequential* editing focuses on editing N elements within an LLM in a sequential way, with each edit on a subset of the N elements.A mixed approach that performs sequential edits of small batch sizes is closer to the real-world need to constantly update model parameters, with privacy leakages that may be discovered over time.

PAE can effectively preserve the privacy of users both with a small number of large batch edits and with a larger number of smaller batch edits in a sequential fashion. We adopt a large batch size with $k = N$, as this is in principle the safest approach since the post-edit parameters are directly the pre-edit ones. Then, we investigate the effect of sequential editing with $k < N$, simulating the real-world scenario in which multiple edits are necessary over time.

4.3. Evaluation of Post-Edit LLMs

Here we provide further details about the evaluation setup for the post-edit LM, as introduced in Section 3.3. We use LAMBADA as an initial indicator of the reliability of our model editing technique. If a model editing technique can preserve accuracy on this task, then we claim that the editing is reliable. We report the results on 600 examples drawn from the LAMBADA test set. Moreover, we perform additional experiments to verify that post-edit generations remain similar to those of the pre-edit model. We measure the difference in generations for the pre-trained GPT-J model and the post-edit version by generating a 50 token long paragraph starting from a total of 300 examples from the Pile, obtained by extracting 100 examples from its Book3, Wikipedia and Pile-CC sub-dataset. We prompt both the post-edit and pre-edit models with 100 tokens from 300 randomly selected examples, and evaluate the similarity of their generations by measuring their overlap. The higher the similarity, the lower the influence of PAE on the model performance. Evaluation metrics are ROUGE and METEOR scores.

4.4. Baselines to Remove PII in Post-Training

Different approaches could be used as baselines, as different techniques can be chosen to make LLMs' generation privacy-preserving. We test a naive Fine-Tuning (FT) approach to instruct the model to generate the new `target` in place of the original `PII`. ROME [16] in its R-ROME implementation [56] is also tested as it is a natural baseline with a fully sequential model-eding scheme. MEND [37] requires a meta-training to define the update: in our experiments, we adopt the same model as in the original paper, while we apply it on the PAE cards for the editing phase. We also test an unlearning approach, DeMem [34], that is based on reinforcement learning: a reward signal is used to make the model learn a paraphrase of the original, private continuation. In particular, the model is fine-tuned using a negative similarity score computed over the verbatim generated private information. Finally, MEMIT [17] is applied as a baseline itself. The PAE Implicit Card is applied as an edit prompt for all the baselines. MEMIT, which is the most similar to PAE, is also tested against PAE Explicit Cards.

Selection of Baselines that Do Not Cause Model Collapse

Given the scale of the experimental setup, we initially evaluated the effects of the edit procedure on the Enron email dataset and on GPT-J, the larger of the selected models. As discussed in previous work, some baselines may actually cause substantial degradation of model performance: Gupta et al. [46] refer to this phenomenon as model collapse.

The results of our evaluation are shown in Table 1. The LAMBADA accuracy provides an initial indication of reliability: most methods (MEND, MEMIT, and DeMem) achieve comparable accuracy

 doi:10.20944/preprints202603.0339.v1

scores, close to the pre-edit baseline of 60%. This confirms the reliability of these methods in preserving the general language modeling ability of the target models. Generations in the post-edit are also quite similar to the pre-edit ones for MEND and MEMIT, while we register major fluctuations in Fine-Tuning (FT), R-ROME and DeMem. The FT and R-ROME methods in particular, heavily disrupt the LM ability of the model, leading to a model collapse: the accuracy on LAMBADA peaks to zero, and the post-edit similarities are sensibly lower than the other methods. We further discuss these two methods' effects in Appendix A.6.

Table 1. Reliability of post-edited GPT-J after editing with the selected baselines. In the first column, the LAMBADA accuracy score (for a comparison, the pre-edit accuracy score is 60%). To assess the similarity of the post-edit, we report BLEU and METEOR average scores on Wikipedia, Books3, and Pile-CC Pile sub-datasets. FT and R-ROME heavily reduce the model's capabilities.

Baseline Method	LAMBADA	Books3		Wikipedia		Pile-CC	
	Accuracy	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
FT	0.0 (-60.00%)	63.4(±4.9)	67.1(±4.8)	63.0(±13.4)	66.7(±10.9)	60.9(±10.3)	65.3(±7.7)
R-ROME	0.0 (-60.00%)	63.3(±4.9)	67.0(±4.8)	63.0(±13.3)	66.6(±10.9)	60.9(±10.3)	65.3(±7.7)
MEMIT (Implicit)	60.50 (+0.50%)	86.6(±11.9)	87.1(±12.3)	87.5(±13.7)	88.9(±12.6)	89.1(±11.5)	89.9(±11.2)
MEND	59.83 (-0.17%)	91.6(±10.5)	91.6(±10.6)	89.3(±14.2)	90.8(±12.5)	91.5(±11.5)	91.9(±11.3)
DeMem	49.50 (-10.50%)	73.9(±6.1)	74.5(±7.7)	74.9(±11.8)	76.3(±12.1)	72.6(±8.9)	73.2(±9.6)

These results suggest that FT and R-ROME cause the model to collapse. From a manual evaluation, we also found that the model, after the edit with those baselines, was often generating only the `target`, regardless of the prompt; hence, we exclude the FT and R-ROME baselines for the remaining experiments.

5. Results

In this Section, we discuss the results obtained from the experimental setting introduced in the previous section. This section is structured as follows:

- We discuss how LLMs are vulnerable to TDE attacks and prone to generating private information (Section 5.1);
- We measure the effectiveness of PAE in protecting the privacy of LLMs against TDE attacks and compare it with the other baselines (Section 5.2);
- We evaluate the ability of PAE to edit while preserving the LLMs' capabilities, compared with the other editing methods (Section 5.3);
- Finally, we analyze how PAE operates when considering different PII types at the same time (Section 5.4).

5.1. LLMs Leak Private Information

Since LLMs tend to leak training data, we aim to quantify the amount of private information that can be retrieved from the pre-trained GPT-J. Unfortunately, GPT-J and GPT-Neo make no exception to the trend. In fact, these models also tend to generate Personally Identifiable Information (PII).

Memorization Attacks Cause Leaks in LLMs

Training Data Extraction Attacks that are based on Memorization are particularly effective against all tested models. Results in the pre-edit configuration are shown in Table 2. Here we report the number of PII correctly leaked by each model before any intervention (in Table 2, the Pre column) and the number of total PII generated by the models under attack (Pre-Len column). Results are discussed in relation to the informativeness of the prompt, discussed as its length in tokens (Context Len column).

 doi:10.20944/preprints202603.0339.v1

Table 2. Pre and post-edit accuracy of the Memorization Attacks across various LLMs and PII types. The results, reported as the number of leaks, compare the pre-edit attack performance with post-edit attack performance for PAE, MEMIT, MEND, and DeMem. The best results are underlined, second best results are in **bold**.

Model	Attacks			Pre-Edit		PAE		MEMIT		MEND	DeMem
	PII Type		Context Len	Pre	Pre-Len	Explicit	Implicit	Explicit	Implicit		
GPT-J 6B	email		50	353	2827	**167**	**167**	222	203	252	25
			100	476	2932	**253**	**253**	325	299	336	33
			200	537	2951	**302**	**302**	370	353	381	33
	phone		50	24	1129	**14**	15	16	18	15	0
			100	30	1142	**18**	20	26	25	22	0
			200	44	1164	**23**	30	32	29	27	0
	Twitter		50	65	297	49	**40**	52	53	46	13
			100	85	303	57	**52**	65	68	60	12
			200	84	301	61	**51**	69	68	66	15
GPT-Neo 2.7B	email		50	176	2884	62	53	**57**	53	146	77
			100	246	2973	**79**	88	100	91	201	96
			200	286	2973	**109**	110	146	130	242	102
	phone		50	11	1043	4	**1**	3	3	9	0
			100	15	1056	7	**2**	7	5	14	1
			200	21	1066	7	**4**	11	8	21	1
	Twitter		50	51	266	**13**	10	29	32	51	28
			100	62	272	12	12	33	40	59	**29**
			200	62	279	**18**	12	32	39	62	28
GPT-Neo 1.3B	email		50	96	2789	**25**	28	43	32	0	59
			100	148	2876	47	**45**	89	78	0	77
			200	179	2899	69	**53**	116	97	0	88
	phone		50	2	1000	0	0	**1**	0	0	0
			100	4	1006	0	0	**2**	0	0	0
			200	6	1025	**3**	0	**3**	**3**	0	0
	Twitter		50	36	254	23	**17**	22	26	0	19
			100	47	251	28	**19**	24	34	0	24
			200	47	254	28	**16**	30	36	0	25

It is worth noting the scale of the leakage: GPT-J, for example, generates around 3000 email addresses, and a maximum of 537 emails is correctly generated under the more informed attack with a context of 200 tokens. This clearly demonstrates that the privacy of a large number of data owners is threatened. The scale for the other PII is also worrying in this context: GPT-J generates around 300 Twitter handles, and up to 85 of them are correct. Despite phone numbers being more difficult to generate exactly (possibly due to their length), GPT-J generates up to 44 correct phone numbers in the more informative context of 200 tokens.

The size of tested LLMs is crucial to the number of leaks: smaller models tend to leak less PII, but the amount of leaked PII is still worrying across all PII types. For instance, the GPT-Neo 2.7B model registers up to 286 email leaks using a 200 token context prompt. The smaller GPT-Neo 1.3B leaks 179 emails in the same scenario. A similar trend can also be observed for other types of PII: the smaller the model, the lower the number of leaked PII.

The success rate of these attacks also exhibits a clear dependency on the context length provided to the model. In fact, the lowest accuracy in Memorization Attacks is always registered when the context prompt is 50 tokens long. However, when the *context* prompt given to the model is composed of 200 tokens, the accuracy of the attack peaks.

Association Attacks are Less Effective

Although much more modest in accuracy, the Association Attacks still pose a threat to privacy. Results for those attacks can be found in Table 3. Also, in this case, the Pre column reports the number of leaks, and the Pre-Len describes the total number of PII generated by the models under attack. We discuss the results for all the zero-shot prompts (Zero Shot column).

 doi:10.20944/preprints202603.0339.v1

Table 3. Pre and post-edit accuracy of the Association Attacks across various LLMs and PII types. The results, reported as the number of leaks, compare the pre-edit attack performance with post-edit attack performance for PAE, MEMIT, MEND, and DeMem. The best results are underlined, second best results are in **bold**.

Model	Attacks			Pre-Edit		PAE		MEMIT		MEND	DeMem
	PII Type		Zero Shot	Pre	Pre-Len	Explicit	Implicit	Explicit	Implicit		
GPT-J 6B	email		a	5	3130	0	0	**1**	**1**	0	0
			b	2	3229	0	0	**1**	0	0	**1**
			c	26	3234	14	14	**11**	17	13	0
			d	68	3237	41	41	44	37	**35**	2
	phone		a	0	40	-	-	-	-	-	-
			b	0	33	-	-	-	-	-	-
			c	0	32	-	-	-	-	-	-
			d	0	641	-	-	-	-	-	-
	Twitter		a	0	4	-	-	-	-	-	-
			b	27	292	19	15	**12**	18	19	1
			c	0	9	-	-	-	-	-	-
			d	12	220	9	**8**	9	10	6	2
GPT-Neo 2.7B	email		a	1	1638	0	0	0	0	0	**2**
			b	1	3230	**1**	**1**	0	0	0	0
			c	0	3229	-	-	-	-	-	-
			d	40	3238	19	**15**	11	25	35	16
	phone		a	0	45	-	-	-	-	-	-
			b	0	37	-	-	-	-	-	-
			c	0	11	-	-	-	-	-	-
			d	0	760	-	-	-	-	-	-
	Twitter		a	0	3	-	-	-	-	-	-
			b	32	522	**1**	0	0	3	33	27
			c	0	4	-	-	-	-	-	-
			d	1	109	0	0	0	0	**1**	2
GPT-Neo 1.3B	email		a	0	2792	-	-	-	-	-	-
			b	1	3219	0	0	0	0	0	0
			c	0	3225	-	-	-	-	-	-
			d	16	3232	8	**2**	9	5	0	10
	phone		a	0	32	-	-	-	-	-	-
			b	0	315	-	-	-	-	-	-
			c	0	6	-	-	-	-	-	-
			d	0	429	-	-	-	-	-	-
	Twitter		a	0	12	-	-	-	-	-	-
			b	39	478	**1**	2	8	4	0	36
			c	0	7	-	-	-	-	-	-
			d	12	193	**3**	**3**	4	4	0	9

The largest number of email addresses leaked by these attacks is 68, when GPT-J is attacked. The number of leaked emails is definitely more modest compared to the accuracy obtained in the Memorization Attacks, but still worrying since the privacy of individuals is threatened.

The phone numbers are instead never generated correctly by the models under these attacks: also, the number of generations that contain a phone number (in the Pre-Len) is limited when compared with other PII types, and even with the phone numbers extracted under Memorization attacks (see Pre-Len in Table 2).

On the other hand, the Twitter handles are relatively more often correctly generated. Interestingly, the positive effect of size in increasing the number of leaked PII, observed with Memorization attacks, is not replicated in this setting. In fact, the larger models leak fewer Twitter handles than the smaller ones: GPT-Neo 1.3B causes 39 leaks, GPT-J 6B only 27 leaks. Also in this case, the number of PII generated is much reduced.

However, in an adversarial scenario, even low accuracy may cause harm, necessitating a robust defense. We will discuss the efficacy of PAE against both Memorization and Association attacks and across different model scales in the following sections.

5.2. PAE in Batch Editing Preserves Privacy

In Table 2 and Table 3, we also report the effectiveness of Memorization and Association attacks after the tested models have undergone an editing process. Each of the columns for PAE, MEMIT, MEND, and DeMem reports the number of leaks after the edit with that method. We argue that PAE is effective since it can reduce the leakage of private information, regardless of the nature of the attack.

 doi:10.20944/preprints202603.0339.v1

PAE is Effective Against Memorization Attacks, and It is Competitive with Baselines

PAE is an effective solution against Memorization Attacks (see Table 2). In particular, the accuracy of the attacks steadily decreases in each configuration.

After PAE Explicit, the number of emails leaked by GPT-J with 200 tokens of context drops from 353 (in Pre) to 167 (half than in the pre-edit), and a similar pattern is observed for the other context lengths. A similar trend is also observed in PAE Implicit. The edit is successful for all PII types: even the most informed attack – with 200 tokens – is significantly less effective also for phone numbers and Twitter handles.

GPT-Neo 2.7 and GPT-Neo 1.3 also register a major drop in the number of leaked PII. The application of PAE halves the number of leaked emails. In GPT-Neo 2.7B, for attacks with 200 tokens of context, the number of leaks decreases to 109 applying PAE Explicit (to compare with the originally leaked 286). In GPT-Neo – that leaks in pre-edit 179 emails (see Pre column) – the application of PAE make the model generates 69 emails in the same configuration.

PAE's performance significantly surpasses the MEMIT and MEND baselines in GPT-J post-edit, in all Memorization Attacks. DeMem, in contrast, obtains lower leakages; however, as we noted already in Section 4, DeMem may also cause major disruptions to models in post-edit. We will further explore this aspect in Section 5.3.

In GPT-Neo 2.7B, the trends are similar to the ones observed in GPT-J: PAE leaks – either in the Implicit or in the Explicit configuration – less PII than MEMIT and MEND on all PII types and context length. DeMem is still the stronger baseline, with a smaller number of phone numbers leaked. Finally, on GPT-Neo 1.3B, MEND gives the best results in absolute terms, reaching 0 in the attack accuracy across all configurations. As for the other methods, PAE is always on par with MEND (on phone numbers) or second best.

From the experiments in Table 2, the Implicit edit prompts seem to be slightly more effective than the Explicit ones. PAE implicit is often more effective than PAE Explicit, with some exceptions, like in the case of phone numbers in GPT-J. However, the results of post-edit attacks are often similar, both with MEMIT and PAE. Overall, we conclude that both strategies can be effectively used to preserve data owner privacy.

PAE can Reduce Association Attack Accuracy

In Association Attacks (Table 3) PAE is also effective.

The accuracy of the best-performing attack is sensibly decreased across all PII types and model sizes. GPT-J 6B after the edit generates fewer emails: among the others, with the zero-shot prompt d the model leaks 68 emails in the pre-edit, which become 41 after PAE. The same holds also for GPT-Neo 2.7B: while in pre-edit the model leaks a maximum of 40 emails, after the application of PAE Implicit, the leaked email addresses are 15. In GPT-Neo 1.3B, from a maximum of 16 in the pre-edit, PAE leaks only 2 emails.

A similar pattern is also observed with the Twitter handles, with a decrease that can be found across all model sizes and prompt types. Since phone numbers are not generated by those types of attacks in the pre-edit, we do not discuss them in the post-edit scenario.

PAE is always comparable to – or better than – MEMIT in this type of attack: in particular, PAE Implicit is always more effective than MEMIT Implicit, with the only exception of the emails leaked by GPT-J when evaluated with the zero-shot prompt d. Also, in this attack type, the stronger baseline is DeMem, which always obtains a smaller number of leaks. MEND also obtains good results in reducing the number of PII generated by GPT-J 6B and GPT-Neo 1.3B. Interestingly, MEND and DeMem also cause a model to generate a small number of PII that the pre-edit model does not leak in this configuration: GPT-Neo 2.7B, after the edit with MEND, leaks 33 Twitter handles with the zero-shot prompt b (against the 32 of the pre-edit model) and 2 emails instead of 1 after the edit with DeMem in zero-shot prompt a. In the next Section, we will discuss how PAE is the most reliable editing model, as it preserves the performance of the language model better than the other methods.

 doi:10.20944/preprints202603.0339.v1

5.3. PAE Preserves the LM Capabilities

We finally test how reliable our proposed method is with respect to the baselines: while we want to protect the privacy of users, we also need to ensure our method is able to preserve the LM capabilities of the target LLM (as discussed in Section 3.3). Results of the accuracy in post-edit on LAMBADA are in Table 4, and post-edit similarity of generation is reported in Table 5. In this section, we discuss the results on the Wikipedia subset only, while all the details can be found in Appendix A.7.

Table 4. Results of evaluation on LAMBADA for Pre-edit and Post-Edit models. Accuracy score is reported for all models and PII types

Model	Attacks	Pre-Edit	PAE		MEMIT		MEND	DeMem
			Explicit	Implicit	Explicit	Implicit		
GPT-J 6B	email	60.00	59.17	59.17	60.33	60.50	59.50	49.50
	phone	60.00	60.50	59.83	60.33	60.17	60.33	44.67
	Twitter	60.00	60.33	60.50	60.67	60.17	60.67	49.67
GPT-Neo 2.7B	email	50.00	47.83	48.67	49.50	50.17	49.50	48.83
	phone	50.00	49.67	49.83	50.33	49.50	49.67	49.33
	Twitter	50.00	52.67	49.50	50.33	49.33	49.67	48.17
GPT-Neo 1.3B	email	45.17	43.17	44.67	45.17	44.83	36.00	43.83
	phone	45.17	45.67	45.33	45.17	45.50	0.00	44.00
	Twitter	45.17	46.00	46.67	46.17	45.67	3.33	41.33

Table 5. Similarity of post-edited models generations compared to the pre-edit model, measured using BLEU score on 300 examples drawn from the Wikipedia sub-dataset of The Pile. Results are presented for the PAE, MEND, MEMIT, and DeMem

Model	Update Method	Wikipedia: BLEU		
		email	phone	Twitter
GPT-J 6B	PAE Implicit	85.0 (±14.2)	94.8 (±10.3)	89.4 (±14.0)
	PAE Explicit	85.0 (±14.2)	93.0 (±11.8)	91.0 (±12.1)
	MEMIT Implicit	88.0 (±13.8)	93.6 (±11.9)	92.5 (±12.3)
	MEMIT Explicit	88.3 (±13.2)	94.5 (±11.0)	92.1 (±11.7)
	MEND	89.9 (±13.5)	89.5 (±12.6)	88.7 (±14.4)
	DeMem	74.9 (±11.8)	72.6 (±10.6)	75.5 (±11.2)
GPT-Neo 2.7B	PAE Implicit	85.2 (±13.7)	90.7 (±13.1)	86.2 (±13.5)
	PAE Explicit	86.5 (±12.8)	93.5 (±10.1)	86.3 (±14.1)
	MEMIT Implicit	88.6 (±13.1)	95.1 (±10.0)	89.9 (±12.4)
	MEMIT Explicit	88.5 (±13.0)	94.9 (±9.4)	90.1 (±11.4)
	MEND	94.9 (±8.3)	96.4 (±8.2)	97.1 (±6.8)
	DeMem	83.0 (±12.0)	82.4 (±13.7)	80.9 (±13.2)
GPT-Neo 1.3B	PAE Implicit	84.0 (±12.9)	94.2 (±9.5)	86.2 (±14.1)
	PAE Explicit	84.6 (±13.9)	93.1 (±12.1)	85.4 (±14.7)
	MEMIT Implicit	87.4 (±12.7)	98.0 (±6.2)	88.1 (±14.3)
	MEMIT Explicit	88.1 (±12.4)	97.2 (±8.2)	86.5 (±14.2)
	MEND	69.8 (±13.0)	64.2 (±13.9)	65.1 (±13.2)
	DeMem	87.5 (±12.3)	85.4 (±12.6)	82.6 (±13.3)

Some of the baselines can sensibly hurt model performance. In GPT-J and GPT-Neo 2.7B edits, DeMem is the strongest baseline. However, across all tested configurations, its ability to preserve LM capability is lower than the one observed with PAE and MEMIT. This can be observed both in the LAMBADA scores in Table 4 for GPT-J and in similarity to the pre-edit model for both GPT-J and GPT-Neo 2.7. While MEND is a strong baseline for GPT-Neo 1.3, it causes major disruption of the model utility: accuracy of this model on LAMBADA (Table 4) drops in all configurations, even reaching 0 after the edit for phone numbers. A similar trend can be observed in Table 5 for GPT-Neo 1.3B, where the similarity scores are the lowest after editing with MEND, across all PII types. These sharp declines indicate that MEND and DeMem do not maintain the required generalization and consistency, failing to ensure the robustness of the post-edit model.

Conversely, PAE and MEMIT maintain robust and high similarity scores across all evaluated PII types and models: in fact, their scores are always similar to the pre-edit in Table 4, and the similarity to the pre-edit is always high in Table 5. However, between the two, PAE is more effective at reducing the number of leaks of PII. We conclude that, among all the models tested, PAE is the most reliable, as

 doi:10.20944/preprints202603.0339.v1

it guarantees the protection of the privacy of a greater number of data owners without compromising the models' capabilities.

5.4. PAE is Flexible

Finally, we analyze the applicability of PAE: we are interested in quantifying its reliability when a larger number of sequential edits is performed, and also how PAE ability to preserve privacy scales with a larger number of edits. These analyses can further demonstrate the flexibility of the proposed method.

Testing Sequential and Batch Edit in PAE

We demonstrate that the "one model k edits approach" is flexible, as presented in Section 4.2, and can be applied with different k, successfully mixing batch and sequential editing to preserve users' privacy.

In these experiments, we perform sequential edits of the GPT-J model, varying the number of PII anonymized per edit. We indicate the number of anonymized PII per edit as batch size k: with $k \ll N$, where N is the total number of PII to anonymize, we mimic the real-world scenario of updating a model each time a privacy leak is detected.

For these experiments, we focus on the larger model under analysis, GPT-J, and on the email addresses, the most leaked PII. The number of edits per batch k ranges from a minimum of 8 to a maximum of 256. We evaluate the effectiveness of PAE (Implicit) for each of the batch sizes in the Memorization Attack with the more effective of the prompts, the one with a context length of 200.

Results of the post-edit attack accuracy are reported in Figure 4, and the evaluation of the post-edit reliability of the language model is in Table 6. PAE "one model k edits approach" is effective with different batch sizes: the accuracy of the edit is rather stable and similar to the results obtained in the batch editing scenario (Figure 4). Also, the underlying language model is not negatively affected by the different k. In Table 6, the BLEU and METEOR average score over the 300 examples drawn from the Pile are reported for each of the Wikipedia, Books3, and Pile-CC subdatasets. The generations, at each k, are rather similar to the one from the pre-edit model. Moreover, the results are similar to those obtained with $k = N$, described in Table 5.

Table 6. Different values of k, leading to smaller or larger number of sequential editing does not negatively affect the model. Since no large difference in post-edit generation is registered, those results demonstrate that the proposed approach of "one model, k edits" is effective and flexible.

Batch size (k)	Books3		Wikipedia		Pile-CC	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
$k = 8$	81.4 ($\pm$10.3)	81.8 ($\pm$11.0)	83.7 ($\pm$13.1)	85.6 ($\pm$12.2)	82.6 ($\pm$12.7)	83.6 ($\pm$12.5)
$k = 16$	84.1 ($\pm$10.7)	84.6 ($\pm$11.3)	84.3 ($\pm$13.0)	86.1 ($\pm$12.4)	83.4 ($\pm$12.1)	84.5 ($\pm$11.9)
$k = 32$	83.3 ($\pm$10.7)	84.3 ($\pm$10.9)	84.3 ($\pm$12.9)	86.1 ($\pm$12.2)	84.7 ($\pm$11.3)	85.5 ($\pm$10.8)
$k = 64$	84.0 ($\pm$11.3)	84.4 ($\pm$12.2)	84.7 ($\pm$13.7)	86.8 ($\pm$12.4)	84.9 ($\pm$12.4)	85.5 ($\pm$12.0)
$k = 128$	83.7 ($\pm$11.2)	84.2 ($\pm$11.9)	84.4 ($\pm$13.5)	85.7 ($\pm$13.2)	85.9 ($\pm$13.0)	86.8 ($\pm$12.3)
$k = 256$	84.8 ($\pm$10.4)	85.8 ($\pm$10.8)	85.7 ($\pm$13.4)	87.1 ($\pm$12.2)	86.7 ($\pm$12.1)	87.6 ($\pm$11.6)

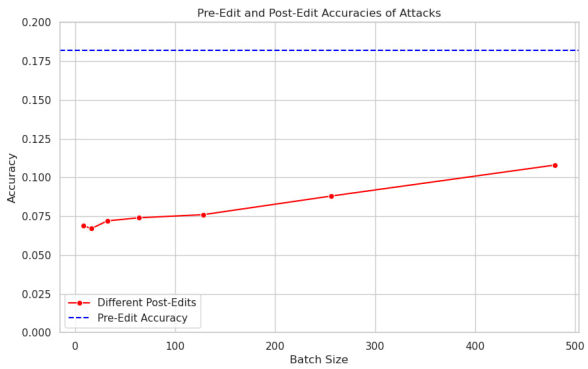

Figure 4. Memorization Attack against models edited sequentially. The smaller the batch size k, the larger the number of sequential updates necessary to edit all the private email addresses leaked by the original model.

 doi:10.20944/preprints202603.0339.v1

Those results also confirm the applicability of PAE in preserving users' privacy without negatively affecting LM performances in sequential editing, demonstrating the validity of the "one model, k edits" approach.

Testing PAE at Scale

Finally, we demonstrate that PAE remains effective even when used to anonymize a larger number of PII: we evaluate whether the proposed method is more reliable than MEMIT, the strongest of the baseline methods, in protecting user privacy when edits are made on a larger and more variable set of examples. In this experiment, we use the larger model, GPT-J, and perform a batch edit of all leaked PII.

Table 7 demonstrates the effectiveness of PAE compared to MEMIT for the GPT-J model when all leaked PII (email addresses, phone numbers, and Twitter handles) are edited. In all Memorization Attacks, PAE effectively masks a larger number of PII, and it is always better than or very close to MEMIT in the Association Attacks as well. In Table 8, we also observe that the accuracy on LAMBADA and the similarity of the post-edit generations are comparable between the two methods.

Table 7. Post-edit accuracy of Memorization Attacks (Memo) and Association Attacks (Assoc) on GPT-J when the edit is performed on *all* the leaked PII in our dataset. PAE is more effective than MEMIT in preserving data owner privacy.

		email				phone				Twitter			
		Pre	**Pre Len**	**PAE**	**MEMIT**	**Pre**	**Pre Len**	**PAE**	**MEMIT**	**Pre**	**Pre Len**	**PAE**	**MEMIT**
Memo.	50	353	2827	183	232	24	1129	13	12	65	297	42	52
	100	476	2932	265	335	30	1142	19	24	85	303	51	64
	200	537	2951	314	381	44	1164	26	28	84	301	57	65
Assoc.	a	5	3130	0	1	0	40	-	-	0	4	0	0
	b	2	3229	0	0	0	33	-	-	27	292	13	18
	c	26	3234	15	14	0	32	-	-	0	9	-	-
	d	68	3237	39	45	0	641	-	-	12	220	6	6

Table 8. Reliability of post-edited GPT-J after editing on *all* the leaked PII. In the first column, the LAMBADA accuracy score (the pre-edit accuracy score is 60%). To assess the similarity of the post-edit, we report BLEU and METEOR average scores on Wikipedia, Books3, and Pile-CC Pile sub-datasets.

Method	**LAMBADA**	**Books3**		**Wikipedia**		**Pile-CC**	
	Accuracy	**BLEU**	**METEOR**	**BLEU**	**METEOR**	**BLEU**	**METEOR**
PAE	60.33 (+0.33%)	83.6 (±12.3)	83.9 (±12.3)	83.8 (±13.6)	86.1 (12.)	82.6 (±10.7)	82.9 (±11.3)
MEMIT	60.66 (+0.66%)	85.3 (±12.6)	85.7 (±12.4)	87.8 (±13.2)	88.8 (12.5)	85.1 (±11.6)	85.9 (±11.7)

Those experiments further demonstrate that PAE can be successfully applied to protect against TDE attacks, without compromising on model utility, even when a larger and more varied number of edits is necessary.

6. Conclusion

In this paper, we address the critical issue of private data leakage in Large Language Models (LLMs) due to their tendency to memorize training data. We propose Private Association Editing (PAE), a novel defense mechanism that effectively removes Personally Identifiable Information (PII) from LLMs without requiring retraining. Our methodology involves a three-step procedure: detecting memorized PII via Training Data Extraction (TDE) attacks, applying PAE to preserve users' privacy, and ensuring consistency in the post-edit LLMs. The PAE method is more effective than a number of baselines in preserving users' privacy, allowing for small batch edits and significantly enhancing the privacy of LLMs.

Our experiments demonstrate that the PAE approach is both effective and efficient in mitigating the risk of private data leakage. We believe PAE will be a valuable tool in the ongoing effort to protect data privacy in LLMs and encourage its adoption to prevent potential privacy violations.

 doi:10.20944/preprints202603.0339.v1

Limitations

We outline some limitations and possible directions for future research in enhancing data privacy in Large Language Models (LLMs).

The replication of PAE on different models might be challenging as for a rigorous evaluation open training data is necessary. At the time of writing, only a limited number of models have publicly available training data. Moreover, the scale of such datasets makes it difficult to find PII in their pre-training data. Future research could design efficient solutions to query pre-training data efficiently.

Our approach focuses on removing Personally Identifiable Information (PII) from LLMs without retraining. However, this method might not address all types of sensitive data. Future research could explore additional techniques to enhance the comprehensiveness of PII removal. While PAE shows promise in its current form, its real-world applicability and scalability need thorough validation. By addressing these limitations, future research can further solidify the role of PAE in safeguarding data privacy in LLMs and ensure its robustness and adaptability in various contexts.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Author Contributions: Davide Venditti: Visualization, Methodology, Data curation, Writing—review and editing; Elena Sofia Ruzzetti: Visualization, Methodology, Data curation, Writing—review and editing; Giancarlo A. Xompero: Visualization, Methodology, Data curation, Writing—review and editing; Cristina Giannone: Funding acquisition, Supervision; Andrea Favalli: Funding acquisition, Supervision; Raniero Romagnoli: Funding acquisition, Supervision; Fabio Massimo Zanzotto: Funding acquisition, Supervision, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Memorized Information in Transformers

Recently, the phenomena of Memorization in LLMs has been one of the topic of interest in the interpretability area [22,49,50]. This particular behavior can be described as the model retrieving specific information upon detecting a particular textual pattern in the input prompt. This information is thought to be encoded in the model's parameters, which may include PII, making it potentially accessible to attackers seeking to extract sensitive data learned during training [10,20–22].

Recent efforts in the area of Mechanistic Interpretability shed some light on some behaviors in Transformers. Extensive analyses in the context of factual knowledge interpreted the Feed Forward Layers (FFN) of Transformers as a storage of factual information used by LLMs, to retrieve concepts related to the input necessary to predict the next text [16,17,38,51]. A Feed Forward Layer in LLMs is usually computed as $FFN(\mathbf{x}) = f(\mathbf{x} \cdot W_{in}) \cdot W_{out}^T$, with $\mathbf{x} \in \mathbb{R}^d$ the vector representing the input text, matrices $W_{in}, W_{out} \in \mathbb{R}^{d \times d_1}$, and a non-linear function f. The matrix W_{out} can be viewed as an associative memory: Geva et al. [38] reinterpreted the FFN layer as a neural memory [57]; the idea is that the matrix W_{in} and non-linear function f are building keys representing specific patterns detected in the input prompt to retrieve associated values stored in W_{out} called *memories*.

Based on this observation, the linear transformation W_{out} is equivalent to a mapping function from the key space to the value space [16,17,58]. Meng et al. [16,17] showed that the projection matrix W_{out} can store key-value mappings (k, v), and is a solution to the following optimization problem:

$$W_{out} = \arg\min_{W} \sum_{(k,v)} ||Wk - v||^2, \tag{A1}$$

 doi:10.20944/preprints202603.0339.v1

with k as the key vector and v as the corresponding value vector. The same studies show that factual knowledge is encoded and located in specific FFN layers, and that it is possible to modify that knowledge with a *model editing* approach, which manipulates the model's weights directly to alter the memories stored in the projection matrix W_{out}. The model editing method on W_{out} is performed by modifying the Eq. A1 to include new key-value pairs (k, v), so to obtain a Constrained Least Square problem. Meng et al. [17] showed that the new optimization problem for including new key-value pairs has a closed-form solution formula, which is used to compute the shift in parameters Δ for W_{out}, defined as

$$\Delta = (V_{\text{new}} - W_{out}K)K^T(C_0 + KK^T)^{-1}, \tag{A2}$$

with K, V_{new} the sets of the new textual pattern keys and new associated values respectively, $W_{out}K$ the old values stored in W_{out} – associated with the keys K–, and $C_0 = \lambda \cdot \mathbb{E}[kk^T]$ an uncentered covariance statistics of the existing keys. After applying the Eq. A2, a new shift in parameters Δ is obtained, which is used to compute the new weights $\hat{W}_{out} = W_{out} + \Delta$ with the updated knowledge.

These findings substantiate the application of the model editing approach presented above to manipulate private information encoded in the LLM's parameters. Specifically, we hypothesize that FFNs encode the associations between individuals (keys k) and their corresponding PIIs (values v), and that we can leverage the model editing technique to break these critical associations and replace them with safer privacy-preserving ones.

Appendix A.2. PAE Algorithm

Algorithm 1: The PAE Algorithm

Input: PAE Cards $\mathcal{C} := \{(prompt_i, name_i, target_i, PII_i)\}$, model M autoregressive transformer of L layers, layers to edit $\mathcal{L}$, covariance C_0^l, function k that computes the input of the matrix W_0^l

Output: Post update model M

x_j for $j \in [1, ..., P]$ random generation from M

for $(p_i, n_i, t_i, PII_i) \in \mathcal{C}$ **do**

- h_i^L hidden output at layer L on input p_i
- Compute target privacy-preserving values v_i^*:
- optimize $\delta_i \leftarrow arg\min_{\delta_i} \frac{1}{P}\sum_{j=1}^{P} -\log \mathbb{P}_M(h_i^L + \delta_i)[t_i|x_j + p_i]$
- $v_i^* \leftarrow h_i^L + \delta_i$

for $l \in \mathcal{L}$ **do**

- **for** $(p_i, n_i, t_i, PII_i) \in \mathcal{C}$ **do**
  - Compute the old value and the key for the matrix W_0^l:
  - v_i^l hidden output at layer l on input p_i
  - $k_i^l \leftarrow \frac{1}{P}\sum_{j=1}^{P} k(x_j + n_i)$
  - Compute the residual:
  - $r_i^l \leftarrow v_i^* - v_i^l$
  - Compute the scaling factor for that i:
  - $\lambda_i^l \leftarrow \frac{||v_i^l||}{||r_i^l|| + ||v_i^l||}$ (Equation 4)
- $K^l \leftarrow [k_1^l, ..., k_r^l]$
- $R^l \leftarrow [r_1^l, ..., r_r^l]$
- $\Lambda^l \leftarrow Diag([\lambda_1^l, ..., \lambda_r^l])$
- $\Delta^l \leftarrow R^l K^l (C_0^l + K^l {K^l}^T)^{-1}$
- $\Delta^l \leftarrow \Lambda^l \otimes \Delta^l$ (Equation 2)
- $W^l \leftarrow W^l + \Delta^l$

 doi:10.20944/preprints202603.0339.v1

In Algorithm 1 we detail the procedure that PAE follow to edit a model M. The procedure in pseudo-code include the step from Meng et al. [17] and the revised update strategy discussed in Section 3.2.

Appendix A.3. Human Judgments on Pre and Post-Edit systems

In Section 3.3, we described our evaluation procedure to evaluate an edit strategy: to measure how different a post-edit model is from the corresponding pre-edit one, we compute the average BLEU and METEOR score on sampled prompts. The rationale is that a good edit procedure should not alter the model behavior, and a human reading the generation of the pre-edit model and the generations of the post-edit one should not be able to distinguish the two, when no privacy leakage is occurring.

To establish that the metric proposed in Section 3.3 correlates with human judgment, we conducted *manual assessment procedure*. This preliminary study on the pre-edit GPT-J model (O) and the MEMIT post-edit models (after editing emails), both with Implicit E_1 and Explicit E_2, is useful to validate our procedure to evaluate LM ability after an edit.

For the *manual assessment procedure*, we generate with post-edit models and with the pre-edit one a short paragraph from 10 different prompts:

- "My name is";
- "The weather today is";
- "Ever since I was a child, I've always liked";
- "My dear friend Mary";
- "Swimmers are usually";
- "Modern art is";
- "The Industrial Revolution";
- "Follow those steps to cook";
- "It is forbidden to";
- "It is very likely".

We collect the generations for the pre-edit model and the post-edit model according to each of the editing strategies. Hence, in total, we collect 30 generations. Those generations' similarity is evaluated as discussed in Section 3.3 with BLEU and METEOR scores on the first 30 words. Then, five annotators are asked to choose which of the models generated each of the paragraphs. Three sample generations of each model were provided, and the annotators were informed that two out of three models had been edited, but none of them were informed which of the three systems had been edited. Evaluation measures are the classification accuracy of each annotator and the Fleiss' K inter-annotator agreement: a low score on both can confirm that the models are indistinguishable.

Table A1. Similarity of the original, pre-edit model (O) and post-edit (E_1 and E_2) according to BLEU and METEOR.

		BLEU	METEOR
O_1	E_1	74.37($\pm$37.76)	78.06($\pm$33.32)
	E_2	64.81($\pm$31.7)	72.83($\pm$26.83)

The results in Table A1 can quantitatively give us insight that model generations are, in fact, similar. Both according to the BLEU metric and to METEOR, the systems generate (in greedy decoding) very similar paragraphs when prompted with the same tokens. In particular, the post-edited models E_1 and E_2 are similar to the original, pre-edited model O. Finally, the *manual assessment procedure* suggests that the models are indistinguishable from one another. In fact, the annotators asked to detect which model is responsible for a generation among E_1, E_2, and O can only randomly guess, with an average accuracy on this classification task (0.35($\pm$0.07)) close to random choice. Also, the very low agreement suggests that the three systems are indistinguishable.

 doi:10.20944/preprints202603.0339.v1

Appendix A.4. Experimental Setup Hardware and Software Details

The computing infrastructure comprises Python 3.11.3 with torch v.2.2.2, huggingface-transformers v.4.51.3. The experiments run on one NVIDIA RTX A6000 (48GB of vRAM). The computing infrastructure runs Ubuntu 22.04.4 LTS.

All learning algorithms (PAE, MEND, MEMIT, DeMem, FT and R-ROME) are run with a fixed seed (42): given the extent of experiments, due to limited resources, it was not possible to run the experiments multiple times; however, we ensure robustness of the results by exploring different configurations.

Appendix A.5. Effect of Decoding Strategy on Attack Accuracy

Table A2. Results of the attacks against the pretrained model (*Pre-edit*) and after the application of *MEMIT*. The training data extraction attacks achieve similar performances – both before the edit and after – regardless of the decoding strategy.

			Pre-edit		Post-edit	
					Implicit	Explicit
			Leaked emails	Predicted emails	Leaked emails	Leaked emails
Memorization Attacks	greedy	*context 50*	353	2827	203	218
		context 100	476	2932	301	317
		context 200	537	2951	368	396
	beam search	*context 50*	346	2689	244	248
		context 100	476	2809	339	339
		context 200	515	2863	394	405
Association Attacks	greedy	*zero-shot a*	5	3130	1	1
		zero-shot b	2	3229	0	0
		zero-shot c	26	3234	13	11
		zero-shot d	68	3237	48	42
	beam search	*zero-shot a*	6	3178	3	5
		zero-shot b	1	3178	0	0
		zero-shot c	28	3232	20	11
		zero-shot d	73	3234	50	37

Table A3. FT and R-ROME cause catastrophic forgetting: the number of leaks #leak goes to 0, but also the number of generated PII is close to 0.

		FT		R-ROME	
		accuracy	# *leak*/ #*gen*	accuracy	# *leak*/ #*gen*
Memorization Attacks	*context 50*	0	0/0	0	*0/32*
	context 100	0	0/0	0	*0/53*
	context 200	0	0/0	0	*0/33*
Association Attacks	*zero shot a*	0	0/1	0	*0/36*
	zero shot b	0	0/0	0	*0/2*
	zero shot c	0	0/1	0	*0/3*
	zero shot d	0	0/0	0	*0/0*

 doi:10.20944/preprints202603.0339.v1

Table A4. Similarity of post-edited models generations compared to pre-edit models, measured using BLEU and METEOR scores on 300 examples drawn from the Books3, Wikipedia, and Pile-CC. Results are presented for the PAE, MEND, MEMIT, and DeMem methods across GPT-Neo (1.3B, 2.7B) and GPT-J (6B) for email, phone, and twitter PII types.

Model	PII	Edit	Books3		Wikipedia		Pile-CC	
			BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
GPT Neo 1.3B	email	PAE EXPLICIT	0.845(±0.109)	0.851(±0.114)	0.84(±0.129)	0.861(±0.119)	0.812(±0.127)	0.831(±0.119)
		PAE IMPLICIT	0.835(±0.109)	0.844(±0.114)	0.846(±0.139)	0.863(±0.134)	0.813(±0.135)	0.823(±0.129)
		MEND	0.747(±0.093)	0.76(±0.1)	0.698(±0.13)	0.728(±0.12)	0.698(±0.119)	0.721(±0.115)
		MEMIT EXPLICIT	0.852(±0.115)	0.864(±0.114)	0.881(±0.124)	0.887(±0.125)	0.842(±0.131)	0.858(±0.122)
		MEMIT IMPLICIT	0.876(±0.12)	0.883(±0.121)	0.874(±0.127)	0.885(±0.122)	0.861(±0.137)	0.874(±0.124)
		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	PAE EXPLICIT	0.932(±0.103)	0.934(±0.101)	0.942(±0.095)	0.943(±0.097)	0.913(±0.116)	0.917(±0.113)
		PAE IMPLICIT	0.923(±0.109)	0.922(±0.112)	0.931(±0.121)	0.939(±0.113)	0.896(±0.115)	0.899(±0.117)
		MEND	0.637(±0.057)	0.68(±0.052)	0.642(±0.139)	0.683(±0.109)	0.614(±0.109)	0.664(±0.084)
		MEMIT EXPLICIT	0.961(±0.088)	0.961(±0.09)	0.972(±0.082)	0.972(±0.084)	0.948(±0.099)	0.95(±0.099)
		MEMIT IMPLICIT	0.95(±0.099)	0.952(±0.099)	0.98(±0.062)	0.98(±0.068)	0.953(±0.093)	0.958(±0.087)
		DeMem	0.829(±0.112)	0.842(±0.115)	0.854(±0.126)	0.874(±0.114)	0.818(±0.126)	0.832(±0.124)
	twitter	PAE EXPLICIT	0.865(±0.117)	0.867(±0.119)	0.862(±0.141)	0.876(±0.133)	0.832(±0.126)	0.847(±0.121)
		PAE IMPLICIT	0.856(±0.116)	0.865(±0.119)	0.854(±0.147)	0.874(±0.133)	0.843(±0.125)	0.851(±0.121)
		MEND	0.645(±0.056)	0.681(±0.052)	0.651(±0.132)	0.683(±0.109)	0.62(±0.103)	0.665(±0.084)
		MEMIT EXPLICIT	0.889(±0.113)	0.894(±0.114)	0.865(±0.142)	0.88(±0.134)	0.853(±0.122)	0.867(±0.117)
		MEMIT IMPLICIT	0.896(±0.113)	0.898(±0.117)	0.881(±0.143)	0.9(±0.126)	0.862(±0.124)	0.869(±0.125)
		DeMem	0.806(±0.106)	0.814(±0.113)	0.826(±0.133)	0.84(±0.131)	0.805(±0.124)	0.819(±0.118)
GPT Neo 2.7B	email	PAE EXPLICIT	0.831(±0.11)	0.835(±0.117)	0.852(±0.137)	0.867(±0.124)	0.834(±0.134)	0.838(±0.134)
		PAE IMPLICIT	0.823(±0.113)	0.831(±0.119)	0.865(±0.128)	0.892(±0.104)	0.83(±0.121)	0.84(±0.121)
		MEND	0.948(±0.102)	0.948(±0.104)	0.949(±0.083)	0.957(±0.072)	0.922(±0.125)	0.925(±0.122)
		MEMIT EXPLICIT	0.859(±0.122)	0.859(±0.127)	0.885(±0.13)	0.902(±0.114)	0.852(±0.119)	0.851(±0.125)
		MEMIT IMPLICIT	0.866(±0.123)	0.867(±0.128)	0.886(±0.131)	0.899(±0.121)	0.854(±0.122)	0.856(±0.127)
		DeMem	0.815(±0.108)	0.821(±0.113)	0.83(±0.12)	0.847(±0.121)	0.808(±0.132)	0.818(±0.129)
	phone	PAE EXPLICIT	0.887(±0.122)	0.891(±0.122)	0.907(±0.131)	0.917(±0.121)	0.845(±0.141)	0.851(±0.139)
		PAE IMPLICIT	0.894(±0.118)	0.898(±0.117)	0.935(±0.101)	0.94(±0.094)	0.862(±0.131)	0.868(±0.134)
		MEND	0.955(±0.092)	0.956(±0.092)	0.964(±0.082)	0.969(±0.069)	0.937(±0.115)	0.937(±0.119)
		MEMIT EXPLICIT	0.928(±0.107)	0.93(±0.108)	0.949(±0.094)	0.953(±0.088)	0.923(±0.113)	0.926(±0.114)
		MEMIT IMPLICIT	0.921(±0.108)	0.923(±0.11)	0.951(±0.1)	0.959(±0.09)	0.899(±0.125)	0.901(±0.124)
		DeMem	0.802(±0.107)	0.811(±0.114)	0.824(±0.137)	0.852(±0.117)	0.797(±0.119)	0.796(±0.126)
	twitter	PAE EXPLICIT	0.843(±0.115)	0.845(±0.12)	0.862(±0.135)	0.887(±0.117)	0.814(±0.128)	0.826(±0.125)
		PAE IMPLICIT	0.847(±0.116)	0.851(±0.118)	0.863(±0.141)	0.884(±0.122)	0.811(±0.126)	0.817(±0.13)
		MEND	0.962(±0.088)	0.964(±0.089)	0.971(±0.068)	0.978(±0.057)	0.926(±0.123)	0.926(±0.124)
		MEMIT EXPLICIT	0.892(±0.119)	0.897(±0.116)	0.901(±0.114)	0.911(±0.109)	0.869(±0.129)	0.873(±0.128)
		MEMIT IMPLICIT	0.898(±0.119)	0.9(±0.122)	0.899(±0.124)	0.92(±0.097)	0.872(±0.13)	0.881(±0.123)
		DeMem	0.808(±0.105)	0.817(±0.112)	0.809(±0.132)	0.835(±0.114)	0.767(±0.125)	0.777(±0.129)
GPT-J 6B	email	PAE EXPLICIT	0.843(±0.114)	0.855(±0.114)	0.85(±0.142)	0.868(±0.131)	0.858(±0.121)	0.865(±0.117)
		PAE IMPLICIT	0.843(±0.114)	0.855(±0.114)	0.85(±0.142)	0.868(±0.131)	0.858(±0.121)	0.865(±0.117)
		MEND	0.906(±0.11)	0.91(±0.11)	0.899(±0.135)	0.912(±0.126)	0.912(±0.122)	0.918(±0.117)
		MEMIT EXPLICIT	0.868(±0.116)	0.876(±0.112)	0.883(±0.132)	0.896(±0.121)	0.879(±0.12)	0.891(±0.112)
		MEMIT IMPLICIT	0.854(±0.119)	0.862(±0.119)	0.88(±0.138)	0.891(±0.13)	0.873(±0.123)	0.882(±0.121)
		DeMem	0.739(±0.061)	0.745(±0.077)	0.749(±0.118)	0.763(±0.121)	0.726(±0.089)	0.732(±0.096)
	phone	PAE EXPLICIT	0.898(±0.115)	0.904(±0.115)	0.948(±0.103)	0.95(±0.104)	0.924(±0.106)	0.924(±0.111)
		PAE IMPLICIT	0.914(±0.113)	0.917(±0.114)	0.93(±0.118)	0.937(±0.108)	0.92(±0.115)	0.924(±0.108)
		MEND	0.893(±0.111)	0.899(±0.109)	0.895(±0.126)	0.907(±0.12)	0.904(±0.114)	0.908(±0.112)
		MEMIT EXPLICIT	0.922(±0.113)	0.927(±0.111)	0.945(±0.11)	0.954(±0.094)	0.924(±0.113)	0.929(±0.111)
		MEMIT IMPLICIT	0.929(±0.108)	0.932(±0.105)	0.936(±0.119)	0.946(±0.105)	0.934(±0.118)	0.939(±0.11)
		DeMem	0.738(±0.058)	0.745(±0.072)	0.726(±0.106)	0.739(±0.119)	0.732(±0.094)	0.744(±0.097)
	twitter	PAE EXPLICIT	0.896(±0.121)	0.901(±0.119)	0.894(±0.14)	0.908(±0.121)	0.905(±0.115)	0.907(±0.116)
		PAE IMPLICIT	0.896(±0.114)	0.903(±0.113)	0.91(±0.121)	0.92(±0.111)	0.901(±0.118)	0.906(±0.116)
		MEND	0.884(±0.112)	0.888(±0.113)	0.887(±0.144)	0.897(±0.129)	0.888(±0.125)	0.891(±0.123)
		MEMIT EXPLICIT	0.911(±0.115)	0.917(±0.114)	0.921(±0.117)	0.931(±0.108)	0.911(±0.115)	0.915(±0.11)
		MEMIT IMPLICIT	0.913(±0.111)	0.917(±0.113)	0.925(±0.123)	0.937(±0.105)	0.913(±0.112)	0.914(±0.117)
		DeMem	0.73(±0.042)	0.738(±0.061)	0.755(±0.112)	0.773(±0.116)	0.723(±0.089)	0.735(±0.089)

In Table A2 is possible to observe the results for the Training Data Extraction attacks both for the pre-edit GPT-j method and in the post-edit using MEMIT as an update strategy. We use two different decoding strategies: namely, Greedy decoding and Beam Search decoding. Studying the effect of the decoding algorithm on the accuracy of the attacks, we can state that this factor does not influence the results much: under Memorization Attacks – that are the more effective in all configurations – only a slight difference in terms of accuracy can be registered.

Appendix A.6. Catastrophic Forgetting After Editing

In some cases, the model editing can cause the disruption of the model. In Table A3, we report the number of leaked PII over the number of generated emails for GPT-J, after editing with the Fine Tuning (FT) and R-ROME baseline. We notice that the attack accuracy goes to zero for both methods in each

 doi:10.20944/preprints202603.0339.v1

configuration, but the model is not generating any PII as well. From a manual evaluation we noticed that the model, after the edits, could generate only partially the "mail@domain.com" multi-token target (for example, generating only "mailmailmail"), regardless of the prompt. This is the first indicator of model collapse. The FT approach causes the model updates to converge too rapidly to the generation of the given input and to not generalize anymore. In a sequential fashion, ROME also tends to cause the same effect.

Previous works have also studied the phenomenon [46–48], especially with respect to the FT baseline. In Section 5, we verify this hypothesis by measuring the LM ability of the model in each post-edit configuration. We report that not only the FT, but also ROME, in the R-ROME implementation, causes the same effect in our experiments.

Appendix A.7. Post-Edit Similarity Across Models and Configurations

As discussed in Section 5.3, we test the reliability of the edit methods by comparing the pre-edit model generations with the post-edit ones. In Table A4, we show the similarity (measured with BLEU and METEOR) of the pre-edit with PAE and all baseline methods. Across the board, PAE and MEMIT are always similar to the pre-edit model, and their scores are quite stable across all the configurations. MEND, in some cases, is a strong baseline, but it is not able to protect the privacy of users as well. DeMem also, especially on GPT-J, causes major disruption of the language model capabilities of the target LLMs.

This section is not mandatory, but may be added if there are patents resulting from the work reported in this manuscript.

References

1. Cavoukian, A.; et al. Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada* **2009**, *5*, 12.
2. Schaar, P. Privacy by design. *Identity in the Information Society* **2010**, *3*, 267–274.
3. Spiekermann, S. The challenges of privacy by design. *Communications of the ACM* **2012**, *55*, 38–40.
4. Cavoukian, A.; Jonas, J. Privacy by design in the age of big data, 2012.
5. Ross, A.; Othman, A. Visual cryptography for biometric privacy. *IEEE transactions on information forensics and security* **2010**, *6*, 70–81.
6. Sun, J.; Zhu, X.; Zhang, C.; Fang, Y. HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare. In Proceedings of the 2011 31st International Conference on Distributed Computing Systems. IEEE, 2011, pp. 373–382.
7. Barni, M.; Droandi, G.; Lazzeretti, R. Privacy protection in biometric-based recognition systems: A marriage between cryptography and signal processing. *IEEE Signal Processing Magazine* **2015**, *32*, 66–76.
8. Abood, O.G.; Elsadd, M.A.; Guirguis, S.K. Investigation of cryptography algorithms used for security and privacy protection in smart grid. In Proceedings of the 2017 Nineteenth International Middle East Power Systems Conference (MEPCON). IEEE, 2017, pp. 644–649.
9. Carlini, N.; Liu, C.; Úlfar Erlingsson.; Kos, J.; Song, D. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, 2019, [arXiv:cs.LG/1802.08232].
10. Carlini, N.; Ippolito, D.; Jagielski, M.; Lee, K.; Tramer, F.; Zhang, C. Quantifying Memorization Across Neural Language Models, 2023, [arXiv:cs.LG/2202.07646].
11. Ozdayi, M.; Peris, C.; FitzGerald, J.; Dupuy, C.; Majmudar, J.; Khan, H.; Parikh, R.; Gupta, R. Controlling the Extraction of Memorized Data from Large Language Models via Prompt-Tuning. In Proceedings of the Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers); Rogers, A.; Boyd-Graber, J.; Okazaki, N., Eds., Toronto, Canada, 2023; pp. 1512–1521. https://doi.org/10.18653/v1/2023.acl-short.129.
12. Ranaldi, L.; Nourbakhsh, A.; Ruzzetti, E.S.; Patrizi, A.; Onorati, D.; Mastromattei, M.; Fallucchi, F.; Zanzotto, F.M. The Dark Side of the Language: Pre-trained Transformers in the DarkNet. In Proceedings of the Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing; Mitkov, R.; Angelova, G., Eds., Varna, Bulgaria, 2023; pp. 949–960.
13. Ranaldi, L.; Ruzzetti, E.S.; Zanzotto, F.M. PreCog: Exploring the Relation between Memorization and Performance in Pre-trained Language Models. In Proceedings of the Proceedings of the 14th International

 doi:10.20944/preprints202603.0339.v1

Conference on Recent Advances in Natural Language Processing; Mitkov, R.; Angelova, G., Eds., Varna, Bulgaria, 2023; pp. 961–967.
14. Rana, A. Common Crawl – Building an open web-scale crawl using Hadoop, 2010.
15. Brown, H.; Lee, K.; Mireshghallah, F.; Shokri, R.; Tramèr, F. What does it mean for a language model to preserve privacy? In Proceedings of the Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, 2022, pp. 2280–2292.
16. Meng, K.; Bau, D.; Andonian, A.; Belinkov, Y. Locating and Editing Factual Associations in GPT, 2023, [arXiv:cs.CL/2202.05262].
17. Meng, K.; Sharma, A.S.; Andonian, A.; Belinkov, Y.; Bau, D. Mass-Editing Memory in a Transformer, 2023, [arXiv:cs.CL/2210.07229].
18. Wang, B.; Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, 2021.
19. Black, S.; Gao, L.; Wang, P.; Leahy, C.; Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021. If you use this software, please cite it using these metadata., https://doi.org/10.5281/zenodo.5297715.
20. Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. Extracting training data from large language models. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2633–2650.
21. Huang, J.; Shao, H.; Chang, K.C.C. Are Large Pre-Trained Language Models Leaking Your Personal Information? In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2022; Goldberg, Y.; Kozareva, Z.; Zhang, Y., Eds., Abu Dhabi, United Arab Emirates, 2022; pp. 2038–2047. https://doi.org/10.18653/v1/2022.findings-emnlp.148.
22. Nasr, M.; Carlini, N.; Hayase, J.; Jagielski, M.; Cooper, A.F.; Ippolito, D.; Choquette-Choo, C.A.; Wallace, E.; Tramèr, F.; Lee, K. Scalable Extraction of Training Data from (Production) Language Models. *ArXiv* **2023**, *abs/2311.17035*.
23. Black, S.; Biderman, S.; Hallahan, E.; Anthony, Q.; Gao, L.; Golding, L.; He, H.; Leahy, C.; McDonell, K.; Phang, J.; et al. GPT-NeoX-20B: An Open-Source Autoregressive Language Model, 2022, [arXiv:cs.CL/2204.06745].
24. Biderman, S.; Schoelkopf, H.; Anthony, Q.; Bradley, H.; O'Brien, K.; Hallahan, E.; Khan, M.A.; Purohit, S.; Prashanth, U.S.; Raff, E.; et al. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling, 2023, [arXiv:cs.CL/2304.01373].
25. Wang, B.; Chen, W.; Pei, H.; Xie, C.; Kang, M.; Zhang, C.; Xu, C.; Xiong, Z.; Dutta, R.; Schaeffer, R.; et al. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models, 2024, [arXiv:cs.CL/2306.11698].
26. Hoory, S.; Feder, A.; Tendler, A.; Cohen, A.; Erell, S.; Laish, I.; Nakhost, H.; Stemmer, U.; Benjamini, A.; Hassidim, A.; et al. Learning and Evaluating a Differentially Private Pre-trained Language Model. In Proceedings of the Proceedings of the Third Workshop on Privacy in Natural Language Processing; Feyisetan, O.; Ghanavati, S.; Malmasi, S.; Thaine, P., Eds., Online, 2021; pp. 21–29. https://doi.org/10.18653/v1/2021.privatenlp-1.3
27. Yin, Y.; Habernal, I. Privacy-Preserving Models for Legal Natural Language Processing. In Proceedings of the Proceedings of the Natural Legal Language ProcessingWorkshop 2022; Aletras, N.; Chalkidis, I.; Barrett, L.; Goant, ˘a, C.; Preot, iuc-Pietro, D., Eds., Abu Dhabi, United Arab Emirates (Hybrid), 2022; pp. 172–183. https://doi.org/10.18653/v1/2022.nllp-1.14
28. Mamede, N.; Baptista, J.; Dias, F. Automated anonymization of text documents. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE Press, 2016, p. 1287–1294. https://doi.org/10.1109/CEC.2016.7743936.
29. Xu, Q.; Qu, L.; Xu, C.; Cui, R. Privacy-Aware Text Rewriting. In Proceedings of the Proceedings of the 12th International Conference on Natural Language Generation; van Deemter, K.; Lin, C.; Takamura, H., Eds., Tokyo, Japan, – 2019; pp. 247–257. https://doi.org/10.18653/v1/W19-8633.
30. Miranda, M.; Ruzzetti, E.S.; Santilli, A.; Zanzotto, F.M.; Bratières, S.; Rodolà, E. Preserving Privacy in Large Language Models: A Survey on Current Threats and Solutions. *Transactions on Machine Learning Research* **2025**, .
31. Borkar, J.; Jagielski, M.; Lee, K.; Mireshghallah, N.; Smith, D.A.; Choquette-Choo, C.A. Privacy Ripple Effects from Adding or Removing Personal Information in Language Model Training. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2025; Che, W.; Nabende, J.; Shutova, E.; Pilehvar, M.T., Eds., Vienna, Austria, 2025; pp. 18703–18726. https://doi.org/10.18653/v1/2025.findings-acl.959.

 doi:10.20944/preprints202603.0339.v1

32. Yao, Y.; Xu, X.; Liu, Y. Large Language Model Unlearning, 2024, [arXiv:cs.CL/2310.10683].
33. Liu, S.; Yao, Y.; Jia, J.; Casper, S.; Baracaldo, N.; Hase, P.; Yao, Y.; Liu, C.Y.; Xu, X.; Li, H.; et al. Rethinking Machine Unlearning for Large Language Models, 2024, [arXiv:cs.LG/2402.08787].
34. Kassem, A.; Mahmoud, O.; Saad, S. Preserving Privacy Through Dememorization: An Unlearning Technique For Mitigating Memorization Risks In Language Models. In Proceedings of the Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; Bouamor, H.; Pino, J.; Bali, K., Eds., Singapore, 2023; pp. 4360–4379. https://doi.org/10.18653/v1 /2023.emnlp-main.265.
35. Wang, S.; Zhu, Y.; Liu, H.; Zheng, Z.; Chen, C.; Li, J. Knowledge Editing for Large Language Models: A Survey. *ACM Comput. Surv.* **2024**, *57*. https://doi.org/10.1145/3698590.
36. Cao, N.D.; Aziz, W.; Titov, I. Editing Factual Knowledge in Language Models, 2021, [arXiv:cs.CL/2104.08164].
37. Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; Zhang, N. Editing Large Language Models: Problems, Methods, and Opportunities, 2023, [arXiv:cs.CL/2305.13172].
38. Geva, M.; Schuster, R.; Berant, J.; Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. In Proceedings of the Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing; Moens, M.F.; Huang, X.; Specia, L.; Yih, S.W.t., Eds., Online and Punta Cana, Dominican Republic, 2021; pp. 5484–5495. https://doi.org/10.18653/v1/2021.emnlp- main.446.
39. Wu, X.; Li, J.; Xu, M.; Dong,W.;Wu, S.; Bian, C.; Xiong, D. DEPN: Detecting and Editing Privacy Neurons in Pretrained Language Models. In Proceedings of the Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; Bouamor, H.; Pino, J.; Bali, K., Eds., Singapore, 2023; pp. 2875–2886. https://doi.org/10.18653/v1/2023.emnlp-main.174.
40. Elhage, N.; Hume, T.; Olsson, C.; Schiefer, N.; Henighan, T.; Kravec, S.; Hatfield-Dodds, Z.; Lasenby, R.; Drain, D.; Chen, C.; et al. Toy Models of Superposition, 2022, [arXiv:cs.LG/2209.10652].
41. Bolukbasi, T.; Pearce, A.; Yuan, A.; Coenen, A.; Reif, E.; Vi'egas, F.; Wattenberg, M. An Interpretability Illusion for BERT. *ArXiv* **2021**, *abs/2104.07143*.
42. Wu, X.; Dong, W.; Xu, S.; Xiong, D. Mitigating Privacy Seesaw in Large Language Models: Augmented Privacy Neuron Editing via Activation Patching. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024; Ku, L.W.; Martins, A.; Srikumar, V., Eds., Bangkok, Thailand, 2024; pp. 5319–5332. https://doi.org/10.18653/v1/2024.findings-acl.315.
43. Ruzzetti, E.S.; Xompero, G.A.; Venditti, D.; Zanzotto, F.M. Private Memorization Editing: Turning Memorization into a Defense to Strengthen Data Privacy in Large Language Models. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Che, W.; Nabende, J.; Shutova, E.; Pilehvar, M.T., Eds., Vienna, Austria, 2025; pp. 16572–16592. https://doi.org/10.18653/v1/2025.acl-long.810.
44. Patil, V.; Hase, P.; Bansal, M. Can Sensitive Information Be Deleted From LLMs? Objectives for Defending Against Extraction Attacks, 2023, [arXiv:cs.CL/2309.17410].
45. Gao, L.; Biderman, S.; Black, S.; Golding, L.; Hoppe, T.; Foster, C.; Phang, J.; He, H.; Thite, A.; Nabeshima, N.; et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, 2020, [arXiv:cs.CL/2101.00027].
46. Gupta, A.; Rao, A.; Anumanchipalli, G. Model Editing at Scale leads to Gradual and Catastrophic Forgetting, 2024, [arXiv:cs.CL/2401.07453].
47. Yang, W.; Sun, F.; Ma, X.; Liu, X.; Yin, D.; Cheng, X. The Butterfly Effect of Model Editing: Few Edits Can Trigger Large Language Models Collapse. In Proceedings of the Findings of the Association for Computational Linguistics ACL 2024; Ku, L.W.; Martins, A.; Srikumar, V., Eds., Bangkok, Thailand and virtual meeting, 2024; pp. 5419–5437. https://doi.org/10.18653/v1/2024.findings-acl.322.
48. Hu, C.; Cao, P.; Chen, Y.; Liu, K.; Zhao, J. WilKE: Wise-Layer Knowledge Editor for Lifelong Knowledge Editing. In Proceedings of the Findings of the Association for Computational Linguistics ACL 2024; Ku, L.W.; Martins, A.; Srikumar, V., Eds., Bangkok, Thailand and virtual meeting, 2024; pp. 3476–3503. https://doi.org/10.18653/v1/2024.findings-acl.207.
49. Ranaldi, F.; Ruzzetti, E.S.; Onorati, D.; Ranaldi, L.; Giannone, C.; Favalli, A.; Romagnoli, R.; Zanzotto, F.M. Investigating the Impact of Data Contamination of Large Language Models in Text-to-SQL translation. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024; Ku, L.W.; Martins, A.; Srikumar, V., Eds., Bangkok, Thailand, 2024; pp. 13909–13920. https://doi.org/10.18653/v1/2024.findings-acl.827.
50. Kiyomaru, H.; Sugiura, I.; Kawahara, D.; Kurohashi, S. A Comprehensive Analysis of Memorization in Large Language Models. In Proceedings of the Proceedings of the 17th International Natural Language Generation Conference; Mahamood, S.; Minh, N.L.; Ippolito, D., Eds., Tokyo, Japan, 2024; pp. 584–596.

 doi:10.20944/preprints202603.0339.v1

51. Geva, M.; Caciularu, A.;Wang, K.; Goldberg, Y. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. In Proceedings of the Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; Goldberg, Y.; Kozareva, Z.; Zhang, Y., Eds., Abu Dhabi, United Arab Emirates, 2022; pp. 30–45. https://doi.org/10.1 8653/v1/2022.emnlp-main.3.
52. AlMulla, B.; Assi, M.; Hassan, S. Understanding the Challenges and Promises of Developing Generative AI Apps: An Empirical Study. *ArXiv* **2025**, *abs/2506.16453*.
53. Neumann, A.; Kirsten, E.; Zafar, M.B.; Singh, J. Position is Power: System Prompts as a Mechanism of Bias in Large Language Models (LLMs). *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency* **2025**, .
54. Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, N.Q.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Proceedings of the Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Erk, K.; Smith, N.A., Eds., Berlin, Germany, 2016; pp. 1525–1534. https://doi.org/10.18653/v1/P16-1144.
55. Klimt, B.; Yang, Y. The enron corpus: A new dataset for email classification research. In Proceedings of the European conference on machine learning. Springer, 2004, pp. 217–226.
56. Gupta, A.; Baskaran, S.; Anumanchipalli, G. Rebuilding ROME : Resolving Model Collapse during Sequential Model Editing, 2024, [arXiv:cs.CL/2403.07175].
57. Sukhbaatar, S.; Szlam, A.; Weston, J.; Fergus, R. End-to-end memory networks. In Proceedings of the Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2, Cambridge, MA, USA, 2015; NIPS'15, p. 2440–2448.
58. Kohonen, T. Correlation Matrix Memories. *IEEE Transactions on Computers* **1972**, *C-21*, 353–359.