Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X
Volume 4, Issue 2, 2026

SELECTIVE NEURON LEVEL KNOWLEDGE UNLEARNING IN PLMS

Hanzla Farooq[1*], Hammad Majeed[2], Ahmad Raza[3]
[*1]Department of Artificial Intelligence & Data Science, FAST (NUCES), Islamabad, Pakistan.
[2]Department of Computer Science, FAST (NUCES), Islamabad, Pakistan.
[3]Department of Artificial Intelligence & Data Science, FAST (NUCES), Islamabad, Pakistan.
[*1]hanzla4887@gmail.com, [2]hammad.majeed@nu.edu.pk, [3]Ahmad.raza@isb.nu.edu.pk

DOI: https://doi.org/10.5281/zenodo.18712947

Keywords
Knowledge Unlearning, Privacy Preservation, Large Language Models, Neuron-Level Editing, Correlation Clustering

Article History
Received on 10 Jan, 2026
Accepted on 15 Feb, 2026
Published on 18 Feb, 2026

Corresponding Author
Hanzla Farooq

Abstract

Large Language Models (LLMs) have achieved state-of-the-art performance across numerous tasks, from text generation to classification. However, their ability to memorize sensitive information during training introduces significant privacy risks, particularly when such data is exposed or misused. Existing approaches such as retraining models from scratch without sensitive data, are computationally expensive and impractical for real-world scenarios. Moreover, current unlearning methods, including gradient-based and shard-based approaches, are limited by high computational costs, performance degradation, and the inability to remove sensitive knowledge. To address these challenges, this study proposes the Selective Neuron-Level Knowledge Unlearning (SNKU) framework, which targets specific neurons responsible for encoding sensitive knowledge within Pre-trained Language Models (PLMs). By employing Correlation Clustering to identify interdependent neuron groups and utilizing Gradient x Activation to rank neuron importance, SNKU selectively edits high-impact neurons. This targeted approach diminishes the influence of sensitive data on model predictions while preserving overall performance, offering an efficient alternative to traditional, more computationally intensive unlearning methods.

1 Introduction

In recent years, major advancements have been made in the field of Natural Language Processing (NLP) since the development of Large Language Models (LLMs) [1]. These LLMs have transformed the capabilities of existing systems into powerful human-language understandable ones, enabling them to better interpret and communicate with unprecedented accuracy and fluency. Over the years, various architectures of LLMs such as BERT [2], LLaMA [3], GPT [4] and T5 [5] have been proposed, each tailored for specific task based on their strength and capabilities. These models are able to achieve state of the art performance in different tasks like text classification, language modeling, language translation, and text summarization. Despite their remarkable contributions, there are some privacy concerns emerging as these models memorize vast amounts of training data, making them vulnerable to privacy attacks. According to Article 17 of the General Data Protection Regulation (GDPR) [6], individuals have the right to request the deletion of personal data, raising concerns about LLMs' ability to retain and potentially leak sensitive information.

In response, companies which have already trained their machine learning models using individual's private data consider only a possible solution, to retrain their models from scratch without the individual's data. While retraining the model would have been feasible if this was a one-time request. However, there are continuous deletion requests from multiple individuals, making this solution computationally expensive and impractical. To tackle such challenges, Machine Unlearning (MU) [7] was introduced, enabling the selective removal of data influence from an already trained model. Depending on the nature of the request, it can be adapted to remove a class, a feature or a specific data point from a model [8]. Since the emergence of machine unlearning, several approaches have been proposed, which are grouped into two broad categories [9]: Exact Unlearning and Approximate Unlearning. Exact unlearning fully removes data through retraining at the expense of high computational cost, while approximate unlearning reduces data influence more efficiently but does not eliminate it entirely [10].

Initially, MU techniques were designed for linear models and later adapted to small-scale deep learning models. However, applying MU to LLMs is significantly more complex due to their large scale, intricate architecture, and billions of parameters. Although methods such as Sharded, Isolated, Sliced, and Aggregated (SISA) training [11] and gradient-based unlearning [12] have shown promise, they face limitations such as performance degradation and high computational cost. Most research focuses on fine-tuning all or a subset of model parameters through multiple forward passes.

Few studies [13, 14], and more recently [15], have worked on neurons to effectively unlearn information. These studies use Gradient-based techniques which assign importance scores to only individual neurons, resulting in no capturing of inter-dependencies between them. However, research [16] has shown that neurons usually work together as a group in order to learn complex concepts. Whereas such methods tend to treat each neuron in isolation, overlooking how neurons interact to represent complex features.

To address this challenge, our study proposes a Selective Neuron-Level Knowledge Unlearning (SNKU) framework, designed to selectively unlearn unwanted knowledge at the neuron level, capturing both individual importance and interconnections, while minimizing performance degradation. Using Correlation Clustering [17] and Gradient x Activation, our approach clusters neurons based on similarity, selects task relevant clusters, ranks neurons based on importance, and modifies their weights to suppress unwanted knowledge. This ensures an efficient and effective unlearning process while addressing key limitations in existing methods.

The major contributions of our study are summarized as follows:

- Utilizes Correlation Clustering to group neurons based on their interdependencies and selecting task-relevant

clusters, enabling precise selection of task-relevant neurons.

- Utilizes Gradient x Activation to identify the most influential neurons, refining the selection process of clusters and ranking them accordingly based on their importance.
- Modifies top K neurons corresponding weights through gradual decay with noise strategy to efficiently reduce unwanted knowledge influence while minimizing overall performance loss.

2 Related Work

Existing early approaches, such as SISA [11] and gradient-based techniques [12] have demonstrated significant progress in unlearning information. SISA shows efficacy by partitioning the dataset into shards, but it is primarily applicable to deep learning models and lacks scalability to LLMs due to their large architecture. Gradient-based methods are effective in unlearning, but they often lead to performance degradation on retained knowledge as they impact the model parameters. Recent advancements such as Knowledge Gap Alignment (KGA) [18] and Efficient Unlearning (EUL) [19] have focused on addressing the challenge of high computational cost in unlearning. In contrast, in-content unlearning-based methods [20] aim to alter the behavior of black-box LLMs during inference without modifying their parameters. While they are effective and efficient, these approaches merely suppress the manifestation of specific information instead of genuinely removing it from the model.

Another line of work focuses on knowledge attribution such as [13, 14], which leverage Integrated Gradients to identify and modify neurons encoding sensitive information. These techniques are one of the few that focuses on targeting specific neurons rather than parameters. Integrated Gradients [21] basically compute gradients by interpolating along a straight-line path in the input space between actual input and baseline. However, this approach does not capture the relationship between neurons as previous studies [16, 22] have highlighted that neurons also interact as a group to learn complex features. This incapability of the attribution-based methods results in incomplete identification of important neurons for the model's output and may lead towards significant degradation of performance upon modifying them.

3 Methodology

This study proposes SNKU framework that utilizes Correlation Clustering and Gradient x Activation to remove knowledge by suppressing selective neuron activation values.

The methodology is divided into four steps: Neuron Attribution, Correlation Clustering, Cluster Selection, and Weight Modification. Fig 1 illustrates the overall workflow of the proposed Selective Neuron-Level Knowledge Unlearning (SNKU) framework.

Given a batch of input sequences $\mathcal{X}$, the model's objective is to predict the masked token y. Let Y represent the set of possible predictions, $\mathcal{X}$ denotes the input token sequences, and $\boldsymbol{\theta}$ be the model parameters. The probability of predicting Y, formulated as $P(Y \mid \mathcal{X}, \boldsymbol{\theta})$, is computed as the product of the conditional probabilities of each individual token:

$$P(Y|X,\theta) = \prod_{i=1}^{|Y|} P(y_i|X,\theta) \quad \#(1)$$

For unlearning, the model is first fine-tuned on a dataset to acquire task-specific knowledge. Prior work [23] has shown that Feed-Forward Networks (FFNs) operate as key–value memories, where each key correlates with textual patterns in the training data and each value represents a distribution over the vocabulary. During inference, neuron activations are extracted from each FFN layer to identify and analyze their contribution to learned representations.

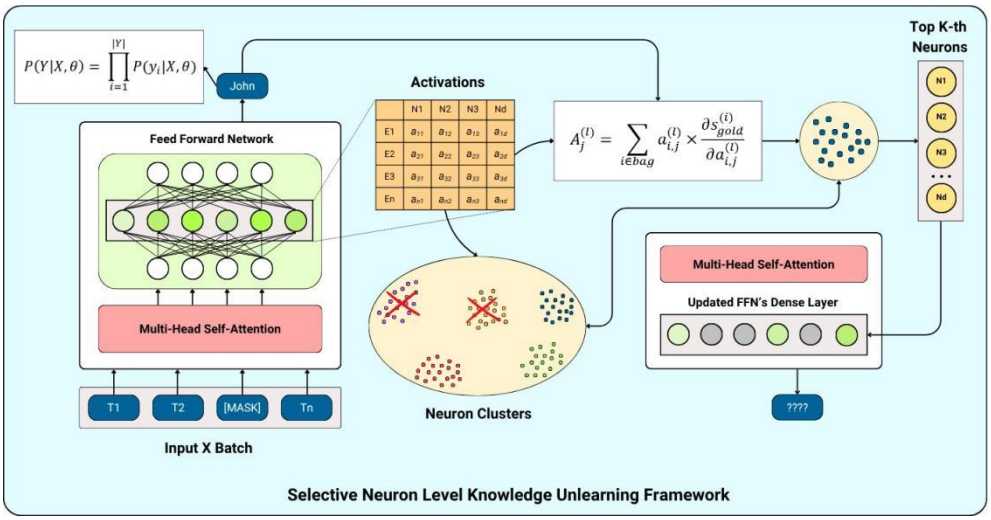

Fig. 1 SNKU extracts neuron activations, computes neuron attributions, and clusters them based on correlation. It then selects task-relevant clusters, ranks their neurons by importance scores, and modifies the top-ranked neuron weights using a gradual-decay with noise mechanism to suppress sensitive knowledge with minimal performance degradation.

3.1 Neuron Attribution

In the Neuron Attribution stage, we quantify how much each FFN neuron in BERT contributes to predicting the masked token. Prior work [13] has shown that editing a small subset of FFN neurons can have a significant impact in altering or removing facts, biases or privacy leaks. To target such neurons effectively with minimal computational cost, we underwent the process of applying Gradient × Activation, a gradient-based method similar to Gradient × Input. Concretely, for each example in a bag B of private sentences, we perform a single forward pass to extract the post-GELU activation vector.

$$a^{(\ell)} \in \mathbb{R}^{d_{ff}} \#(2)$$

at layer ℓ. We then backpropagate the logit score s_{gold} of the gold (true) token to obtain the gradient $\nabla_a(\ell)\, s_{\text{gold}}$. By taking the element-wise product,

$$g^{(\ell)} = \nabla_{a^{(\ell)}} s_{gold} \odot a^{(\ell)} \#(3)$$

we obtain a simple per-neuron proxy for importance in that single example. Finally, we sum these gradient × activation values over all i examples in the bag:

$$A_j^{(\ell)} = \sum_{i \in B} g_{ij}^{(\ell)}, \quad j = 1, \ldots d_{ff} \#(4)$$

This yields, for each layer, a single attribution vector $A^{(\ell)}$ whose entries highlight the neurons that exert the greatest overall influence on predicting the private masked tokens.

3.2 Correlation Clustering

To effectively capture interdependencies between neurons and group them based on functional similarity, we perform global correlation clustering for each layer ℓ. First, we collect the post-GELU activation vectors $a^{(\ell)} \in R^{d}ff$ from each masked token example, stacking them into a matrix:

$$X^{(\ell)} = \begin{bmatrix} a_1^{(\ell)} \\ a_2^{(\ell)} \\ \cdot \\ \cdot \\ \cdot \\ a_N^{(\ell)} \end{bmatrix} \in \mathbb{R}^{N \times d_{ff}}. \#(5)$$

Next, treating each neuron as a point in the N-dimensional example space, we compute the pairwise cosine-similarity matrix $S^{(\ell)}$. The cosine similarity is performed between neuron activation vectors to measure the strength of their linear relationship. Its value lies in the range [−1, 1], where high positive values indicate strong similar relations, high negative values indicate strong opposite relations and values equal to zero indicate no relation at all. We then

convert the resultant matrix $S^{(\ell)}$ into a distance matrix $D^{(\ell)} = 1 - S^{(\ell)}$.

Finally, we apply agglomerative clustering to $D^{(\ell)}$ with a fixed distance threshold of 0.7, following prior redundancy analyses in BERT [24], where this value yielded stable and interpretable neuron groups. By grouping neurons with highly similar activation patterns, we obtain consistent and interpretable clusters for subsequent unlearning steps.

3.3 Cluster Selection

Once neurons are grouped into clusters, the next step is to identify and select the most relevant clusters for unlearning. To achieve this, we follow a simple statistical scoring approach that highlights cluster importance. For each cluster $C_{\ell,c}$ in layer ℓ, we first compute two base statistics from the pre-neuron attributions $A^{(\ell)}$ (as defined in Eq. 4):

the total importance

$$SUM_{l,c} = \sum_{j \in C_{l,c}} A_j^{(l)}, \#(6)$$

and the mean importance

$$AVG_{\ell,c} = \frac{1}{|C_{\ell,c}|} \sum_{j \in C_{\ell,c}} A_j^{(\ell)}, \#(7)$$

Because SUM alone biases towards large clusters and AVG alone favors small ones, we introduce a hybrid score that balances both:

$$HYB_{\ell,c} = \frac{SUM_{\ell,c}}{|C_{\ell,c}|^{\alpha}}, \alpha = 0.5 \#(8)$$

This choice of α provides a balance between SUM and AVG contributions, and we provide an empirical comparison in Appendix C. Next, to filter out weak or spurious groups, we discard any cluster whose average importance falls below 2% of the bag's maximum single-neuron score:

$$AVG_{\ell,c} \geq 0.02 \max_j A_j^{(\ell)}. \#(9)$$

Finally, we rank the remaining clusters by $HYB_{\ell,c}$ and based on an ablation study balancing unlearning effectiveness and computational cost, select the top M = 20 clusters. These top M clusters and the neurons they contain, are the ones we target for modification, ensuring that we focus solely on those groups that are both strongly activated and of reasonable size.

3.4 Weight Modification

Once we have identified the top M clusters, we run a lightweight Python utility to reformat the selection and obtain the set of unique neurons for the weight-editing step. We then sort these unique neurons in descending order according to their attribution scores, ensuring that higher-impact neurons are prioritized for modification. In the weight modification phase, rather than simply zeroing out each selected neuron or replacing it with a default "unknown" embedding, we apply a gradual decay combined with a small amount of Gaussian noise. Concretely, for each privacy category τ (e.g. phone numbers, person names, random text), we choose a decay hyperparameter α_τ and a noise scale σ_τ tailored to that category's structure and sensitivity. Enumerating the K selected neurons

$$(\ell_k, p_k)_{(k=1)}^{K} \quad , \text{ let } w_{(\ell_k,p_k)}^{(old)} \in \mathbb{R}^{(d_{ff})}$$

as be the original weight column of the k-th neuron in layer ℓ_k. We compute a decay factor.

$$\delta_k = 1 - \alpha_\tau^k, \#(10)$$

sample noise

$$\varepsilon_k \sim N(0, \sigma_\tau^2 I), \#(11)$$

and set new weights

$$w_{\ell_k,p_k}^{new} = \delta_k w_{\ell_k,p_k}^{old} + \varepsilon_k \#(12)$$

All modifications occur under a no-gradient context so as not to perturb other learned parameters. By letting the decay base α_τ and noise level σ_τ to vary across privacy categories, we softly erase private signals in a way that respects each category's unique characteristics, while preserving the model's overall performance. Experiments In this section, we present the experimental setup used to evaluate the effectiveness of the proposed unlearning approach. We scribe the dataset, models, and establish baseline comparisons to analyze the results for assessing the impact of our framework.

3.5 Dataset

To ensure that the model memorizes information and to enable fair comparison with prior work, we fine-tune it on the Enron dataset [25]. The Enron dataset is considered to contain private information, as it consists of employee emails, but it was made publicly available by the Federal Energy Regulatory Commission. It

contains over 5,000 real emails, making it the largest publicly available email dataset to date. For the sampling phase, we follow the same strategy as DEPN, categorizing private information into the following types: *Name*, *Tel*, and *Random*. Since this is a masked language modeling task, the tokens to be predicted are masked in each sentence.

3.6 Model Configuration

For unlearning, we use two BERT variants: **BERT-Base**, which comprises 12 transformer layers with a hidden state size of 768 and an intermediate (FFN) size of 3072, and **BERT-Large**, which comprises 24 transformer layers with a hidden state size of 1024 and an intermediate size of 4096. Both variants are fine-tuned on the Enron MLM task for 10 epochs using an NVIDIA RTX 4060 GPU.

3.7 Baseline

To assess the effectiveness of our proposed unlearning method, we compare it against four baselines: FTM, GARE, DEPN and APNEAP. The Fine-Tuned Model (FTM) refers to a BERT model fine-tuned for 10 epochs on Enron dataset without applying unlearning, representing strong utility but high privacy risk. Gradient Ascent for Random Editing (GARE) is a gradient ascent-based baseline where k randomly selected neurons are fine-tuned by maximizing the MLM objective, on memorized samples for a fixed number of epochs. Further implementation details of GARE are given in Appendix B. Detecting and Editing Privacy Neurons (DEPN) consists of three components: the Privacy Neuron Detector computes attribution scores for all neurons using Integrated Gradients; the Privacy Neuron Aggregator filters and selects the top k neurons; and the Privacy Neuron Editor reduces their influence by zeroing out the corresponding weights.

Augmented Privacy Neuron Editing via Activation Patching (APNEAP) extends DEPN by replacing weight zeroing with activation patching. It builds steering vectors, which are activation-difference signals derived by contrasting private inputs with their desensitized counterparts. The steering vectors capture privacy-related activation shifts and are injected during inference to override sensitive activations. The full APNEAP mechanism is detailed in appendix D. These baselines provide a reference to evaluate whether our method achieves effective unlearning while preserving overall model performance.

3.8 Evaluation Metrics

For evaluation, we consider three metrics: Perplexity, Exposure, and Mean Reciprocal Rank (MRR). **Perplexity** is a standard metric in language modeling that measures how well a model predicts a sequence of tokens. Lower values indicate greater confidence in predictions, while higher values reflect uncertainty and reduced utility. **Exposure** measures how easily a private token sequence t of length m can be extracted from a model with parameters θ. It is defined as

$$e_\theta(t) = \log_2 |\mathcal{R}| - \log_2 \left(\text{rank}_\theta(t)\right),$$

where R is the set of all candidate tokens at each position and $\text{rank}_\theta(t)$ is the rank of t when model likelihoods (under θ) are sorted from highest to lowest. A lower exposure value indicates stronger suppression of the memorized sequence. **Mean Reciprocal Rank (MRR)** evaluates how highly private entities (e.g., names) are ranked in the model's predictions. Let a multi-token entity be $E = \{e_1, e_2, \ldots, e_{|E|}\}$ and Q the prompt context. For each token e_i, let $\text{rank}_\theta(e_i \mid Q)$ denote its position when model likelihoods (under parameters θ) are sorted from highest to lowest. Then

$$\text{MRR}(E) = \frac{1}{|E|}\sum_{i=1}^{|E|}\frac{1}{\text{rank}_\theta(e_i|Q)}$$

A lower MRR indicates that private tokens are ranked lower in the prediction list, signifying more effective unlearning. **Costing Time** refers to the wall-clock time (in seconds) for neuron selection and weight modification per example, indicating the method's computational efficiency. Details on private and non-private data, identification of memorized samples, and how these metrics are applied in our evaluation are provided in appendix A.

3.9 Main Results

This section presents experimental results, highlighting the effectiveness of our unlearning approach. Tables 1 and 2 report the evaluation results on memorized and validation set samples for BERT-Baseand BERT-Large, demonstrating SNKU's scalability to larger models. We evaluated the method using multiple metrics and compared it against baseline models to assess its impact on knowledge retention and performance degradation

Table 1: ***Comparison of FTM, GARE, DEPN, APNEAP, and SNKU on BERT-Base across privacy categories.***

Privacy Category	Metric	FTM	GARE	DEPN	APNEAP	SNKU
TEL	Valid PPL	**2.16**	2.20	2.22	2.21	2.24
	Exposure	15.78	15.50	15.00	14.60	**13.94**
	Time (sec)	–	**794.56**	2200	2800	939.07
NAME	Valid PPL	**2.16**	3.37	3.69	3.20	2.58
	MRR	0.51	0.35	0.34	0.36	**0.33**
	Time (sec)	–	**16.40**	107.64	130	46.01
RANDOM	Perplexity	4.38	5.04	6.97	7.20	**8.81**
	Time (sec)	–	**13.89**	5054.1	5800	2858.3

Table 2: ***Comparison of FTM, GARE, DEPN, APNEAP, and SNKU on BERT-Large across privacy categories.***

Privacy Category	Metric	FTM	GARE	DEPN	APNEAP	SNKU
TEL	Valid PPL	**4.75**	4.79	4.84	4.83	4.88
	Exposure	15.22	14.01	13.44	13.10	**12.76**
	Time (sec)	–	**1940.8**	6486.4	7200	4316.7
NAME	Valid PPL	**5.26**	6.07	6.31	5.90	5.33
	MRR	0.41	0.40	0.39	**0.38**	**0.38**
	Time (sec)	–	**24.60**	403.75	450	178.06
RANDOM	Perplexity	2.62	3.29	5.48	5.60	**6.79**
	Time (sec)	–	**60.82**	14598	15200	8461.7

3.9.1 Phone Number Category

In the phone number category, our framework SNKU demonstrates superior unlearning performance, significantly reducing exposure with only minimal loss in model utility compared to others. The baseline model, FTM exhibits the lowest perplexity since it undergoes no unlearning process, indicating that it fully retains memorized knowledge. Both DEPN and APNEAP, while lowering exposure to some extent, do not achieve the same level of knowledge suppression as SNKU and incur substantial time cost due to Integrated Gradients. APNEAP improves upon DEPN by achieving slightly lower exposure and better perplexity preservation, but its runtime increases further due to the additional step of constructing and applying steering vectors during activation patching. In contrast, GARE shows limited effectiveness in unlearning, achieving the least reduction in exposure among all methods. Notably, despite also relying on attribution scoring, SNKU runs in less than half the time of DEPN and APNEAP, highlighting its efficiency alongside stronger unlearning.

3.9.2 Name Category

In the name category, the results mirror those observed in phones numbers, with SNKU again achieving the strongest unlearning performance, reflected in the lowest MRR while also

maintaining the lowest perplexity among the unlearning methods. DEPN achieves a comparable level of knowledge suppression but suffers from higher perplexity, indicating greater performance degradation. APNEAP improves upon DEPN by offering a slightly better balance between unlearning and utility, though at the expense of increased runtime.

GARE yields modest unlearning results close to DEPN but benefits from reduced time cost since it bypasses attribution scoring. FTM, as expected, remains the best in terms of perplexity because no unlearning is performed, though it carries the highest privacy risk. In addition, SNKU is significantly faster than both DEPN and APNEAP, completing the unlearning process in less than half their runtime while delivering superior privacy and utility trade-offs.

3.9.3 Random Text Category

In the random text category, perplexity is the only metric, with higher values indicating stronger unlearning effectiveness. SNKU achieves the highest perplexity, demonstrating the strongest suppression of memorized information. APNEAP ranks second, surpassing DEPN in perplexity but at the cost of higher runtime. DEPN follows with the third highest perplexity, while GARE provides only a moderate increase yet remains the fastest due to its simplicity. As expected, FTM shows the lowest level of perplexity, reflecting its purely fine-tuned state without any unlearning. Importantly, SNKU achieves the highest perplexity in nearly half the time required by DEPN and APNEAP, highlighting both effectiveness and efficiency.

4 Analysis

In this section, we examine (1) how different neuron budgets affect the privacy–utility trade-off, and (2) the sensitivity of our method to key hyperparameters.

4.1 Privacy–Utility Trade-off Across Neuron Budgets

Fig 2 and fig 3 together paint a comprehensive picture of how the two unlearning methods named DEPN and SNKU, achieve the balance between privacy (exposure or MRR) and utility (validation perplexity) as we vary the number of edited neurons *K*. In fig 2(a), we observe that SNKU achieves a rapid drop in exposure: editing just 400 Neurons reduces exposure by over 1.5 points, whereas

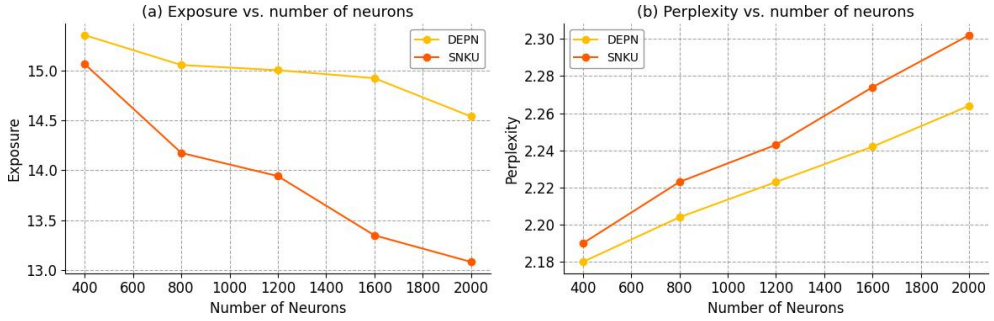

Fig. 2 Comparison between DEPN and SNKU on the TEL category: (a) exposure versus number of edited neurons and (b) validation perplexity versus number of edited neurons.

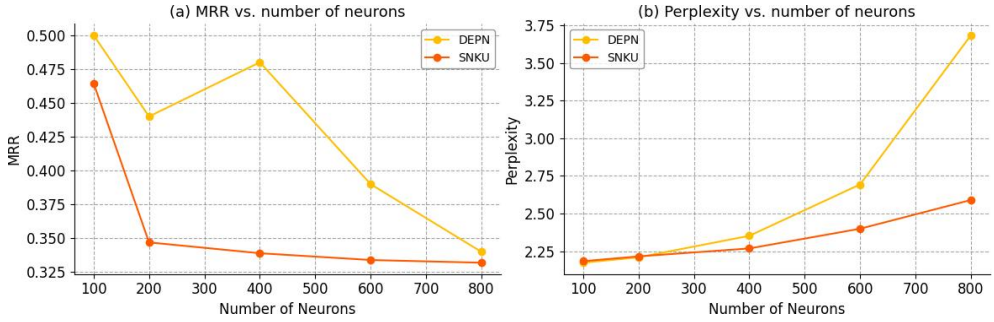

Fig. 3 Comparison between DEPN and SNKU on the NAME category: (a) mean reciprocal rank versus number of edited neurons and (b) validation perplexity versus number of edited neurons.

DEPN's reduction is under 0.5. As SNKU drives exposure down to approximately 13.1, nearly 1.4 points lower than DEPN's 14.6. Meanwhile, fig 2(b) shows that SNKU's perplexity grows more slowly (rising from 2.19 to 2.30) compared to DEPN's 2.18 to 2.26 over the same range. This gap highlights SNKU's ability to remove critical privacy information with minimal disturbance to the model's overall performance.

Turning towards the name category, we see the same pattern in fig 3(a): SNKU slashes the average reciprocal rank from 0.46 down to 0.34 by k = 800, while DEPN only falls from 0.50 to 0.34. That represents a 26% larger drop for SNKU at the highest neuron budget. At the same time in fig 3(b), SNKU's perplexity increases gently from 2.18 to 2.59, whereas DEPN's climbs more sharply to 3.65. In other words, for each additional neuron edited, SNKU delivers a steeper privacy gain per unit of utility loss. Putting both curves side by side, it becomes clear that cluster-based editing targets the most "privacy-heavy" neurons first, yielding strong forgetting with fewer weight perturbations. DEPN's flat ranking of individual neurons, by contrast, requires many more edits to match the same privacy reduction and at a higher perplexity unit. Overall, these analyzes underscore the superior privacy-utility trade-off of SNKU: for any fixed budget k, we consistently see greater reductions in exposure or MRR, paired with smaller increases in perplexity.

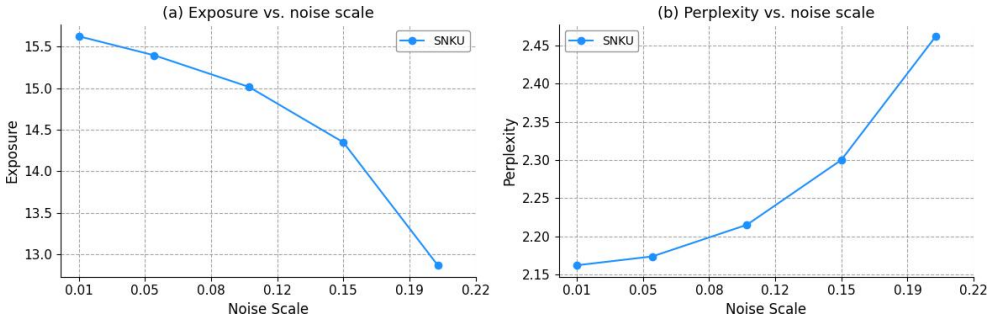

Fig. 4 Noise-scale ablation on the TEL category with a fixed neuron budget k = 300: (a) exposure and (b) validation perplexity as functions of the Gaussian noise scale σ.

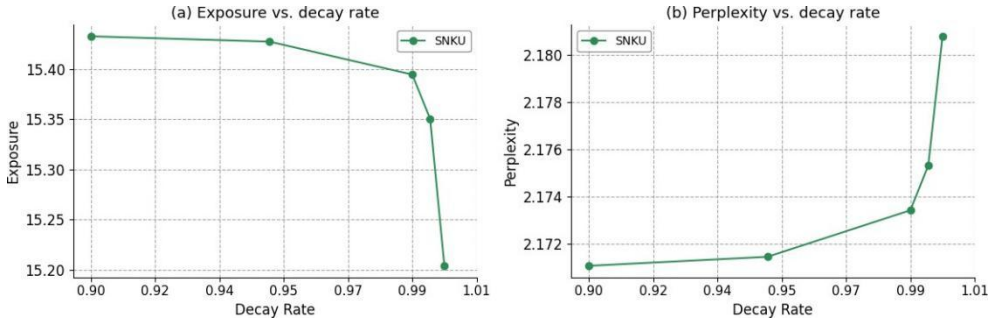

Fig. 5 Decay-rate ablation on the TEL category with a fixed neuron budget k = 300: (a) exposure and (b) validation perplexity as functions of the decay rate α.

4.2 Hyperparameter Sensitivity Analysis

To validate our choice of noise scale and decay rate, we performed two targeted ablations on the TEL category with a fixed neuron budget of k =

300. Fig 4 and fig 5 illustrate how exposure responds to varying each hyperparameter in turn.

4.2.1 Noise Scale (σ) Ablation

With decay fixed at α = 0.99 and k = 300 neurons, increasing the Gaussian noise scale steadily lowers exposure as shown in Fig 4(a). Moving from a very light σ = 0.01 to σ = 0.10 cuts exposure by 0.61 points (15.62→15.01). A further jump to σ = 0.15 yields the single largest drop (-0.67) while keeping the rise in perplexity Fig 4(b)) within an acceptable band. Pushing noise to 0.20 does erase even more (12.87), but in Fig 4(b), the plot shows it also starts to erode downstream utility. Hence σ = 0.15 strikes the best privacy–utility balance for phone numbers.

4.2.2 Decay Rate (α) Ablation

Holding σ = 0.05, we swept α from 0.90 (aggressive decay) up to 0.999 (very mild). In fig 5(a), Exposure falls only slightly between 0.90 and 0.995, then drops more noticeably at α = 0.999 (15.35→15.20). Higher aggressiveness (α < 0.90) offered no additional gain but did raise perplexity shown in Fig 5(b). These results suggest that a slow decay allows the network to adjust gradually and delivers almost all the forgetting benefit while avoiding catastrophic effect in the dense layer.

4.2.3 Setting Hyperparameters

Combining the TEL ablations, we adopt α = 0.9997 and σ = 0.15 as the optimal pair, reducing exposure by ≈ 2.8 points relative to the no-noise baseline while keeping perplexity stable. To ensure reproducibility across categories, we repeated the same search procedure for NAME and RANDOM. For NAME, α = 0.99 and σ = 0.05 achieved the strongest MRR reduction with minimal performance loss. For RANDOM, a moderate setting of α = 0.995 and σ = 0.10 gave the lowest perplexity among the tested pairs. For brevity, we present detailed sensitivity plots only for TEL; summary values for all categories are reported here.

5 Limitations

Despite its advancements, SNKU has two main limitations. First, correlation does not guarantee a causal link to private information when knowledge is represented in distributed or entangled patterns (e.g., multiple neurons jointly encoding overlapping secrets), it can miss crucial neurons or include irrelevant ones, leading to incomplete erasure or unnecessary perturbations. Second, although far cheaper than end-to-end retraining, SNKU still computes attributions for every FFN neuron which can become a major bottleneck as model size grows. Moreover, we were unable to conduct scalability experiments on larger models (e.g., Llama-7B or GPT-models) due to limited computational resources. Future work could apply causal inference to refine neuron selection, explore cross-layer clustering to capture multi-layer interactions, and develop approximate attribution (e.g., neuron sampling) to make unlearning practical for billion-parameter models.

6 Conclusion

In this study, we introduced the Selective Neuron-Level Knowledge Unlearning (SNKU) framework, an unlearning method for targeted knowledge removal in Pre-Trained Language Models (PLMs). Unlike traditional approaches that rely on costly retraining or parameter-wide adjustments, SNKU selectively modifies the activations of neurons that contribute most to the retention of specific knowledge. By leveraging correlation clustering and gradient-based attribution, our approach identifies high-impact neuron groups, ranks them based on importance, and reduces their influence in a structured manner.

Our experimental results demonstrated that SNKU effectively suppresses memorized knowledge while maintaining overall model performance. Comparisons with baseline methods such as FTM, GARE, DEPN and APNEAP confirmed that our selective unlearning strategy minimizes performance degradation while achieving effective knowledge suppression. Furthermore, our approach requires significantly lower computational overhead than retraining-based unlearning methods, making it more scalable for real-world applications.

Ultimately, our work contributes to the growing field of machine unlearning, providing a more efficient, scalable, and targeted solution for privacy preservation and knowledge removal in PLMs.

Acknowledgements

The authors would like to acknowledge the valuable guidance and support provided by the supervisory team throughout the development of this work. Their constructive feedback and continuous encouragement greatly contributed to this study. The authors would also like to thank the National University of Computer and Emerging Sciences for providing the research environment and facilities that supported this work.

Statements and Declarations

Funding

The authors declare that this research was conducted without any external funding.

Competing Interest

The authors declare that they have no competing interests or financial conflicts to disclose.

Appendix A Memorized Sample Identification

Following DEPN, we identify memorized private samples by scoring all private candidates after training and applying DEPN's thresholds: Exposure > 15 for TEL and MRR < 1.5 for NAME. Privacy metrics (Exposure and MRR) are then reported only on this memorized subset, which isolates leakage risk. For non-private data, we use a held-out validation set consisting of 5% of the Enron dataset, following the same splitratio as DEPN.

Under the masked language modeling objective, the model's loss is the average negative log-likelihood (NLL) over masked tokens. Perplexity is derived by exponentiating this average loss, and for multi-token spans (e.g., names or phone numbers), the NLL is averaged across all masked positions in the span. This protocol ensures that unlearning effectiveness is measured on privacy-sensitive cases while overall utility is tracked on general text.

Appendix B GARE Baseline Implementation

For reproducibility, we provide implementation details of the GARE baseline. GARE is realized as a gradient-ascent fine-tuning procedure that maximizes the MLM loss on memorized samples.

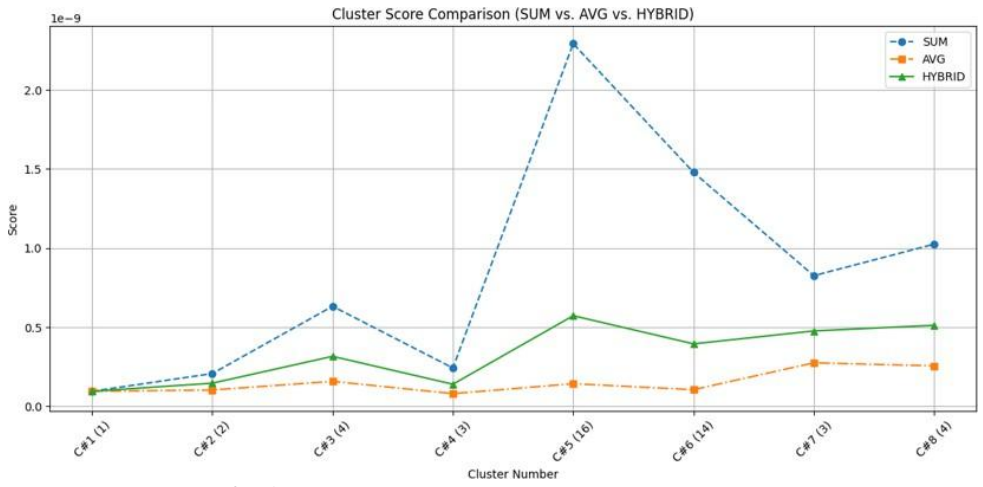

Fig. C1 Comparison of cluster scoring strategies using SUM, AVG, and HYB (α = 0.5). SUM over-emphasizes large clusters, whereas AVG underweights them. HYB provides balanced scoring. X-axis labels (e.g., C#5 (16)) denote the cluster index and the number of neurons per cluster.

At each run, we sample k neurons uniformly at random across all feed-forward layers of BERT. Each neuron corresponds to one output dimension (column) of the FFN dense layers, and only these k positions receive gradient updates; gradients on all other weights are masked to zero.

The value of k is aligned with the comparison baseline in each category. So, for example, if DEPN removes the top 1000 neurons, then GARE also updates k = 1000 randomly selected neurons. Optimization uses AdamW with a learning rate of 5×10^{-5}, 20 epochs, batch size 8, and a linear warmup schedule with 10% of total steps. This setup ensures that GARE reflects a fair gradient-ascent baseline directly comparable to methods that erase or modify neurons.

Appendix C Hybrid Scoring Sensitivity

To validate the choice of α = 0.5 in our HYB scoring function, we compared cluster importance scores produced by SUM, AVG, and HYB across several clusters. Figure C1 shows that SUM tends to over-emphasize larger clusters, while AVG underweights them and flattens the distribution. HYB with α = 0.5 strikes a balance, avoiding the extremes of either method and yielding more stable and interpretable cluster scores across categories. This provides empirical support for fixing α = 0.5 in all experiments.

Appendix D Editing via Activation Patching

APNEAP introduces a novel editing mechanism that steers neuron activations rather than zeroing out weights. The core idea is to modify internal feature representations through steering vectors that push activations away from privacy-encoding directions.

D.1 Construct Desensitized Counterparts

For each private sample (e.g., "*Call me at 912-XXX-1234*"), APNEAP constructs a desensitized counterpart by replacing the sensitive information with a neutral place- holder (e.g., "*Call me at 000-000-0000*"). This pairing helps reveal how privacy information alters neuron activations.

D.2 Compute Steering Vectors

Both versions (sensitive and desensitized) are passed through the model to record neuron activations:

$$H_{sen}, H_{des} \in \mathbb{R}^{n\times m\times d},$$

where n is the number of sample pairs, m is the number of selected privacy neurons, and d is the hidden dimension. The steering vector is then computed as the mean activation difference across all pairs:

$$V = \frac{1}{n}\sum_{i=1}^{n}\left(H_i^{sen} - H_i^{des}\right), \quad V \in \mathbb{R}^{m\times d}.$$

This vector represents the direction in activation space corresponding to the encoding of private information.

D.3 Apply Activation Patching

During inference, APNEAP steers model activations by linearly adding the steering vector:

$$\hat{H} = H + \alpha \cdot V$$

where α is a hyperparameter controlling the patch strength (set to 10 in their experiments). This operation gently nudges the model's activations away from the private direction, effectively neutralizing sensitive signals without altering learned weights.

By combining desensitized sample construction, activation comparison, and steering-based patching, APNEAP effectively erases privacy traces while preserving overall model performance.

References

[1] Kumar, P.: Large language models (llms): survey, technical frameworks, and future challenges. Artif. Intell. Rev. **57**, 260 (2024)

[2] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: North Amer- ican Chapter of the Association for Computational Linguistics (2019). https://api.semanticscholar.org/CorpusID:52967399

[3] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozi`ere, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave,E., Lample, G.: Llama: Open and

efficient foundation language models. ArXiv **abs/2302.13971** (2023)

[4] Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training. (2018). https://api.semanticscholar.org/CorpusID:49313245

[5] Raffel, C., Shazeer, N.M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140–114067 (2019)

[6] Kranenborg, H.: 475 Article 17 Right to erasure ('right to be forgotten'). In: The EU General Data Protection Regulation (GDPR): A Commentary. Oxford Uni- versity Press, ??? (2020). https://doi.org/10.1093/oso/9780198826491.003.0049

[7] Cao, Y., Yang, J.: Towards making systems forget with machine unlearning. In: 2015 IEEE Symposium on Security and Privacy, pp. 463–480 (2015). https://doi.org/10.1109/SP.2015.35

[8] Li, C., Jiang, H., Chen, J., Zhao, Y., Fu, S., Jing, F., Guo, Y.: An overview of machine unlearning. High-Confidence Computing, 100254 (2024) https://doi.org/10.1016/j.hcc.2024.100254

[9] Jiang, Y., Liu, S., Zhao, T., Li, W., Gao, X.: Machine unlearning survey. In: Zhang, D. (ed.) Fifth International Conference on Mechatronics and Computer Technology Engineering (MCTE 2022), vol. 12500, p. 125006. SPIE, ??? (2022). https://doi.org/10.1117/12.2660330 . International Society for Optics and Photonics. https://doi.org/10.1117/12.2660330

[10] Qu, Y., Yuan, X., Ding, M., Ni, W., Rakotoarivelo, T., Smith, D.: Learn to unlearn: Insights into machine unlearning. Computer **57**(3), 79–90 (2024) https://doi.org/10.1109/MC.2023.3333319

[11] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C.A., Jia, H., Travers, A., Zhang, B., Lie, D., Papernot, N.: Machine unlearning. In: 2021 IEEE Symposium on Security and Privacy (SP), pp. 141–159 (2021). https://doi.org/10.1109/SP40001.2021.00019

[12] Jang, J., Yoon, D., Yang, S., Cha, S., Lee, M., Logeswaran, L., Seo, M.: Knowledge unlearning for mitigating privacy risks in language models. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14389–14408. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.acl-long.805 . https://aclanthology.org/2023.acl-long.805

[13] Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., Wei, F.: Knowledge neurons in pretrained transformers. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8493–8502. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-long.581 . https://aclanthology.org/2022.acl-long.581

[14] 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: Bouamor, H., Pino, J.,

Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 2875–2886. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.emnlp-main.174
. https://aclanthology.org/2023.emnlp-main.174

[15] Wu, X., Dong, W., Xu, S., Xiong, D.: Mitigating privacy seesaw in large language models: Augmented privacy neuron editing via activation patching. In: Ku, L.-W., Martins, A., Srikumar, V. (eds.) Findings of the Association for Computational Linguistics: ACL 2024, pp. 5319–5332. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.findings-acl.315 . https://aclanthology.org/2024.findings-acl.315/

[16] McInnes, L., Healy, J., Melville, J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction (2020). https://arxiv.org/abs/1802.03426

[17] Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning **56**, 89–113 (2002)

[18] Wang, L., Chen, T., Yuan, W., Zeng, X., Wong, K.-F., Yin, H.: KGA: A general machine unlearning framework based on knowledge gap alignment. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 13264–13276. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.acl-long.740 . https://aclanthology.org/2023.acl-long.740

[19] Chen, J., Yang, D.: Unlearn what you want to forget: Efficient unlearning for LLMs. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 12041–12052. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.emnlp-main.738 . https://aclanthology.org/2023.emnlp-main.738

[20] Pawelczyk, M., Neel, S., Lakkaraju, H.: In-context unlearning: Language models as few-shot unlearners. In: Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., Berkenkamp, F. (eds.) Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 40034–40050. PMLR, ??? (2024). https://proceedings.mlr.press/v235/pawelczyk24a.html

[21] Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3319– 3328. PMLR, ??? (2017). https://proceedings.mlr.press/v70/sundararajan17a.html

[22] Wu, J., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., Glass, J.: Similar- ity analysis of contextual word representation models. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting

of the Association for Computational Linguistics, pp. 4638–4655. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.422 . https://aclanthology.org/2020.acl-main.422/

[23] Geva, M., Schuster, R., Berant, J., Levy, O.: Transformer feed-forward layers are key-value memories. In: Moens, M.-F., Huang, X., Specia, L., Yih, S.W.-t. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 5484–5495. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021. emnlp-main.446 . https://aclanthology.org/2021.emnlp-main.446/

[24] Dalvi, F., Sajjad, H., Durrani, N., Belinkov, Y.: Analyzing Redundancy in Pretrained Transformer Models (2020). https://arxiv.org/abs/2004.04010

[25] Klimt, B., Yang, Y.: Introducing the enron corpus. (2004)