JU-CSE-NLP'25 at SemEval-2025 Task 4: Learning to Unlearn LLMs

Arkajyoti Naskar, Samitinjaya, Dipankar Das, Sivaji Bandyopadhyay
Jadavpur University, Kolkata, India
{arkajyoti2708, samitgupta03, dipankar.dipnil2005, sivaji.cse.ju}@gmail.com

Abstract

Large Language Models (LLMs) have achieved enormous success recently due to their ability to understand and solve various non-trivial tasks in natural language. However, they have been shown to memorize their training data which, among other concerns, increases the risk of the model regurgitating creative or private content, potentially leading to legal issues for the model developer and/or vendors. Such issues are often discovered post-model training during testing or red teaming. While unlearning has been studied for some time in classification problems, it is still a relatively underdeveloped area of study in LLM research since the latter operates in a potentially unbounded output label space. Specifically, robust evaluation frameworks are lacking to assess the accuracy of these unlearning strategies. In this challenge, we aim to bridge this gap by developing a comprehensive evaluation challenge for unlearning sensitive datasets in LLMs.

1 Introduction

Large Language Model (LLM) unlearning is selectively removing specific knowledge from a trained model while retaining its general capabilities. This is crucial in scenarios where models inadvertently memorize sensitive information, contain outdated or incorrect data, or need to comply with user requests for data removal under regulations like GDPR. Unlike traditional model retraining, which is computationally expensive and impractical for large-scale models, efficient unlearning methods seek to erase targeted knowledge with minimal computational overhead.

Our system employs a two-pronged approach to tackle this challenge: ***Normalized Gradient Difference*** (NGDiff) for selective unlearning and ***AutoLR*** for dynamic learning rate adaptation. NGDiff modifies model parameters by computing the gradient differences between retain and forget datasets, ensuring targeted forgetting while minimizing unintended knowledge loss. This method allows the model to maintain critical learned information while efficiently removing specific instances.

AutoLR enhances this process by optimizing the learning rate through quadratic loss fitting. This allows the system to dynamically adapt its updates for faster convergence and stability. For every 10 iteration, AutoLR evaluates the model's loss behavior with different learning rates and selects the optimal value to ensure stable and effective parameter updates. The combination of NGDiff and AutoLR makes our system an efficient and scalable solution for the problem of selective unlearning.

Through our participation in this task, we observed that applying NGDiff with AutoLR significantly improved unlearning efficiency compared to static learning rate approaches.

However, challenges remained, particularly in handling complex QA examples where forgetting was less effective. Instances involving long-context dependencies or indirect knowledge retrieval were harder to unlearn, suggesting that additional refinements to the gradient normalization process could further enhance performance. Additionally, we noted that aggressively tuning AutoLR in early epochs sometimes led to instability, indicating the need for more adaptive thresholding techniques.

2 Task Description

The challenge covered three sub-tasks spanning different document types:

1. **Subtask 1:** Long-form synthetic creative documents spanning different genres.

2. **Subtask 2:** Short-form synthetic biographies containing *personally identifiable information* (PII), including fake names, phone numbers, SSNs, email, and home addresses.

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 2059–2064

3. **Subtask 3:** Real documents sampled from the target model's training dataset.

For each task,[Ramakrishna et al., 2025b] there were two types of evaluation - Sentence Completion and Question Answering (QA). A trained Large Language Model (LLM) was provided to memorize documents from all three tasks. For each subtask, there were specific **Retain** (i.e., documents the model should retain in memory) and **Forget** sets (i.e., documents the model should forget) along with the target model. The primary task was to develop an algorithm to unlearn the information present in the Forget set without affecting the information present in the Retain set.

3 Related Work

Several unlearning methods have been explored, each addressing different challenges in forgetting specific information while maintaining generalizability.

3.1 Gradient Difference

Gradient difference methods compute the difference between gradients from the retain and forget datasets to selectively adjust model parameters. The core idea is to reduce the model's reliance on forget-set data while ensuring minimal impact on retain-set performance. The standard gradient difference formulation is given by:

$$g_{diff} = g_R - g_F \tag{1}$$

where g_R and g_F are the gradients computed from the retain and forget datasets, respectively. This method updates the model parameters by applying the computed gradient difference, effectively counteracting the influence of the forget set.[Bu et al., 2024]

3.2 Gradient Ascent

Gradient ascent unlearning reverses the learning process by updating model parameters in the opposite direction of gradients computed on the forget set. While effective for aggressive forgetting, this method risks instability and can cause model divergence if not carefully controlled.

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta L_F(\theta_t) \tag{2}$$

where $L_F(\theta_t)$ is the loss on the forget dataset, η is the learning rate, and ∇_θ represents the gradient with respect to model parameters.[Ginart et al., 2019]

3.3 KL Minimization

Kullback-Leibler (KL) divergence minimization unlearning aims at aligning the model's output distribution after unlearning with a desired target distribution. This method is particularly effective in reducing the dependence of the model on forgotten data while preserving generalization.

$$\min_\theta D_{KL}(P_{forget}(x) \parallel P_{model}(x;\theta)) \tag{3}$$

where $D_{KL}(P \parallel Q)$ is the KL divergence between two probability distributions.[Guo et al., 2020]

3.4 Negative Preference Optimization

By modifying the loss function, negative preference optimization enforces lower confidence in specific outputs associated with forgotten information. This method selectively reduces the likelihood of forgotten data appearing in model predictions without significantly affecting other learned knowledge.

$$L_{neg} = -\sum_i w_i \log(1 - P_\theta(y_i|x_i)) \tag{4}$$

where $P_\theta(y_i|x_i)$ is the probability the model assigns to a forgotten instance and w_i is a weighting factor.[Yao et al., 2024]

3.5 Preference Optimization

Preference optimization techniques adjust the model's training objective to explicitly reduce reliance on undesired information while strengthening important knowledge. This approach ensures a structured forgetting process without excessive performance degradation.

$$L_{pref} = \alpha L_{retain} + (1 - \alpha) L_{forget} \tag{5}$$

where α controls the trade-off between forgetting and retention objectives.[Bourtoule et al., 2021]

4 System Overview

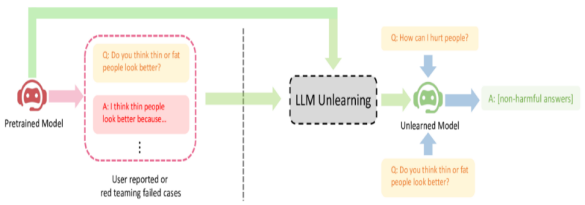

Figure 1: General Block Diagram of LLM Unlearning Process [Yao et al., 2024]

4.1 Datasets

The following datasets have been provided to us:

1. Retain Train Set
2. Forget Train Set
3. Retain Validation Set
4. Forget Validation Set

Figure 1 and Figure 2 give us some insight into the data that was given to us for our task. [Ramakrishna et al., 2025b]

The data sets used were spread across the three subtasks given, the details of which are provided in Section 2. Also, to compare our algorithm with existing algorithms, our task organizers used the ***TOFU*** [1] dataset to run those existing algorithms and provide us with a baseline score of the said algorithms. This dataset comprises question-answer pairs based on autobiographies of 200 different authors that do not exist and are completely fictitiously generated by the GPT-4 [2] model. In addition, during the evaluation phase, along with the data gathered from the organizers, the data set is used to evaluate the algorithm on various evaluation metrics.

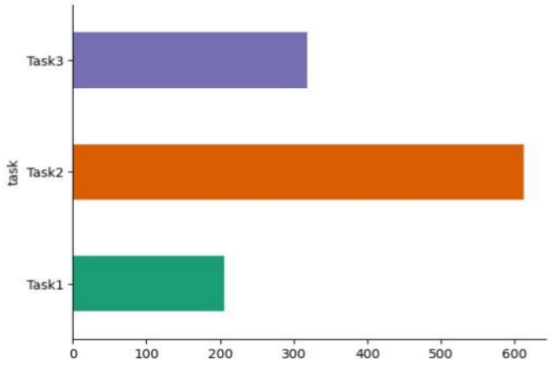

Figure 2: Number of rows of each subtask (Retain Dataset)

[1] https://huggingface.co/datasets/locuslab/TOFU
[2] https://openai.com/index/gpt-4/

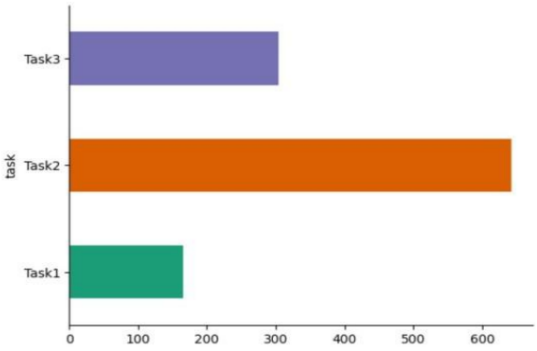

Figure 3: Number of rows of each subtask (Forget Dataset)

The main goal here is to remove selective information present in the forget set from the causal Learning model while preserving the information from the retain set, ensuring that the overall utility of the model is preserved. For this task, an approach called NGDiff Unlearning has been used. This approach is based on computing the gradients separately for the retain and forget sets and then using their difference (NGDiff) to guide the weight updates.

4.2 Custom Dataset Class

The dataset was given in the Pandas data frame format, as discussed above in the Data Set Section. We used a custom dataset class in our algorithm to work with the data. This class takes the pandas dataset given to us, tokenizes the text using the appropriate hugging face tokenizer, and returns the tensor representations of input and output sequences.

Input encodings are generated by tokenizing the input text using the provided tokenizer and converting the tokenized text into a PyTorch tensor using `return_tensors="pt"` during the tokenization process. A similar process has been adopted for the output text. A maximum length of sequence has been set and truncation has been set to true along with setting padding equal to the maximum length of sequence, ensuring uniform sequence length via truncation and padding.

This class returns this tokenized data in the form of PyTorch tensors. `input_ids`, `attention_mask`, `labels` are returned. Here `input_ids` are the tokenized representation of input text. `attention_mask` is a mask that indicates which tokens are real words(1) vs padded tokens(0). Labels are the tokenized representation of output text. While returning these the

`squeeze(0)` operation is applied to each of these to remove the unnecessary batch dimension that has been added because `return_tensors="pt"` adds an extra dimension. This class is used with PyTorch's `Dataloader` to efficiently batch and shuffle data during training.

4.3 Computing Loss and Gradients

An user-defined function named `compute_loss_and_gradients()` has been written that computes the loss and gradients for a given dataset using dataloader. Firstly, it computes the loss and gradients of all the batches in the dataset, and then returns the average loss over all the batches and average gradients accumulated over all the batches. The loss is computed using `outputs.loss` where the output is the output predicted for a given text by the model. `outputs.loss` typically computes the cross-entropy loss if it is not customized or overridden manually. For computing gradients `loss.backward()` is used which computes gradients of loss using model parameters. Gradient clipping is done by capping their norm to `max_norm=1.0`. Gradients are extracted and are accumulated. Normalize the gradients over total batches. The total loss is computed as:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i \tag{6}$$

where N is the number of batches and L_i is the loss for each batch. The gradient accumulation is given by:

$$g = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L_i \tag{7}$$

where g is the averaged gradient over all batches.

4.4 Computing normalized Gradient Difference

We have computed not only the normalized gradients for retain and forget sets but also the difference by subtracting the normalized forget gradients from the normalized retain gradients. The normalized gradient difference is computed as:

$$g_{NGDiff} = \frac{g_R}{||g_R||} - \frac{g_F}{||g_F||} \tag{8}$$

where g_R and g_F are the gradients from the retain and forget datasets and g_{NGDiff} is the normalized gradient difference.[Ramakrishna et al., 2025a]

4.5 Adaptive Learning Rate

The best *learning rate* is dynamically selected to update the model parameters based on the Normalized Gradient Difference(NGDiff). The main idea behind this is to find the optimal learning rate that minimizes the impact of unlearning and preserving the useful knowledge, or in other words the utility of the model.

Quadratic fitting is used to determine the best optimal learning rate. The losses are calculated based on each learning rate. We fit a quadratic function to losses computed for different candidate learning rates and find the minimum:

$$\eta^* = \frac{-b}{2a}, \quad \text{if } a > 0 \tag{9}$$

Otherwise, the minimum learning rate is chosen from predefined candidates:

$$\eta^* = \min(\eta_1, \eta_2, ..., \eta_n) \tag{10}$$

where a and b are the coefficients from quadratic fitting of learning rates and losses. Losses are computed using:

$$L(\eta) = \sum_i |g_R - \eta . g_{NGDiff}|^2 \tag{11}$$

where g_R is the retain gradient, and g_{NGDiff} is the normalized gradient difference. The learning rate that minimizes this function is chosen dynamically.

4.6 LoRA

The ***Low-Rank Adaptation***(LoRA) reduces the number of trainable parameters by decomposing weight updates into low-rank matrices:

$$\Delta W = AB \tag{12}$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$ with $r \ll d, k$. This reduces the computational cost while maintaining effective updates. Since the models that were given, in which unlearning had to be performed, were very large in terms of size. So, running them on low resources and then performing unlearning became a very difficult task. Hence we took advantage of LoRA.

LoRA modifies the fine-tuning process by freezing the original model weights and applying changes to a separate set of weights, which are then added to the original parameters. LoRA transforms the model parameters into a lower-rank dimension, reducing the number of parameters that need training(trainable parameters), thus speeding up the process and lowering costs. A detailed explanation of

how LoRA works and how it is used can be found here [Hu et al., 2021].

5 Evaluation And Results

5.1 Evaluation Metrics

The evaluation metrics that were officially provided by the task organizers are provided below. We have only mentioned the Results that are based on these evaluation metrics. Also, all the three scoring techniques described here were aggregated to form a final score.

1. Task-specific regurgitation rates (measured using ***rouge-L*** scores) on the sentence completion prompts and exact match rate for the question answers on both retain and forget sets. We have inverted the forget set metrics to $1 -$ their value. We have aggregated all 12 distinct scores described above to generate a single numeric score via harmonic mean.

2. A ***Membership Inference Attack*** (MIA) score using loss-based attack on a sample of member and non-member datasets, given by $1 - abs(mia\_normloss\_normauc\_normscore - 0.5) * 2$.

3. The model performance on the ***MMLU*** benchmark, measured as test accuracy on 57 STEM subjects.

5.2 Results

The following models were provided - **7B Model**[3] and **1B Model**[4]. The results were obtained on both of these models separately. Table 1 shows the results obtained by our algorithm by all the scoring techniques that were used by the task organizers.

	Model 7B	Model 1B
Final Score	0.165	0.397
Task Aggregate	0.0	0.0
MIA Score	0.0	0.929
MMLU Score	0.495	0.261
Rank Obtained	11	9

Table 1: Results and Rank obtained by the Unlearning Algorithm for the official Task Evaluation Metrics

[3]https://huggingface.co/allenai/OLMo-7B-0724-Instruct-hf

[4]https://huggingface.co/allenai/OLMo-1B-0724-hf

Analysis of LoRA's Size-Dependent Performance: With a fixed low-rank ratio, LoRA tunes about 48M parameters in the 1B model versus 314M in the 7B model [Hu et al., 2021, Houlsby et al., 2019], that is, roughly 4.8% of each network. However, the 7B version still freezes approximately 6.7B weights, so its updates cover a much smaller fraction of the overall parameter space. According to established scaling laws [Kaplan et al., 2020], fixed ratio adaptations yield diminishing returns at larger scales, which explains why the 1B model achieves stronger targeted forgetting under LoRA.

6 Conclusion

We formulate the machine learning problem and propose a novel NGDiff unlearning method based on the normalized gradient difference and the adaptive learning rate. By leveraging insights from multitask optimization, NGDiff improves forgetting quality while maintaining utility of the Retain set.

Due to limited compute and data, we relied on LoRA (Low-Rank Adaptation) to shrink the number of trainable parameters while preserving performance. We explored several adaptive learning-rate schedules to minimize loss, but extensive hyperparameter sweeps and larger-scale evaluations remain out of reach. In future work, we will investigate alternative parameter-reduction techniques—such as prompt tuning or sparsity-based pruning—and conduct more thorough ablation studies to further improve unlearning efficacy.

References

Ludovic Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Haoran Jia, Nicholas Travers, Bita Zhang, Junjie Zhang, David Evans, and Daniel Hsu. Machine unlearning. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021. URL https://experts.illinois.edu/en/publications/machine-unlearning.

Zhiqi Bu, Xiaomeng Jin, Bhanukiran Vinzamuri, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, and Mingyi Hong. Unlearning as multi-task optimization: A normalized gradient difference approach with an adaptive learning rate, 2024. URL https://arxiv.org/abs/2410.22086.

Jonathan A. Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019. URL https://shorturl.at/p3FYl.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR, 2020. doi: 10.48550/arXiv.1911.03030. URL https://arxiv.org/abs/1911.03030. Accessed: 2025-04-21.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Anil Ramakrishna, Yixin Wan, Xiaomeng Jin, Kai-Wei Chang, Zhiqi Bu, Bhanukiran Vinzamuri, Volkan Cevher, Mingyi Hong, and Rahul Gupta. Lume: Llm unlearning with multitask evaluations. *arXiv preprint arXiv:2502.15097*, 2025a.

Anil Ramakrishna, Yixin Wan, Xiaomeng Jin, Kai-Wei Chang, Zhiqi Bu, Bhanukiran Vinzamuri, Volkan Cevher, Mingyi Hong, and Rahul Gupta. Semeval-2025 task 4: Unlearning sensitive content from large language models. *arXiv preprint*, 2025b.

Yaxuan Wang, Jiaheng Wei, Chris Yuhao Liu, Jinlong Pang, Quan Liu, Ankit Parag Shah, Yujia Bao, Yang Liu, and Wei Wei. Llm unlearning via loss adjustment with only forget data, 2024. URL https://arxiv.org/abs/2410.11143.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning, 2024. URL https://arxiv.org/abs/2310.10683.

[Ramakrishna et al., 2025a] [Ramakrishna et al., 2025b] [Hu et al., 2021] [Wang et al., 2024] [Yao et al., 2024] [Hu et al., 2021, Houlsby et al., 2019] [Kaplan et al., 2020] [Ginart et al., 2019] [Guo et al., 2020] [Bourtoule et al., 2021]