UNIFIED PARAMETER-EFFICIENT UNLEARNING FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The advent of Large Language Models (LLMs) has revolutionized natural language processing, enabling advanced understanding and reasoning capabilities across a variety of tasks. Fine-tuning these models for specific domains, particularly through Parameter-Efficient Fine-Tuning (PEFT) strategies like LoRA, has become a prevalent practice due to its efficiency. However, this raises significant privacy and security concerns, as models may inadvertently retain and disseminate sensitive or undesirable information. To address these issues, we introduce a novel instance-wise unlearning framework, LLMEraser, which systematically categorizes unlearning tasks and applies precise parameter adjustments using influence functions. Unlike traditional unlearning techniques that are often limited in scope and require extensive retraining, LLMEraser is designed to handle a broad spectrum of unlearning tasks without compromising model performance. Extensive experiments on benchmark datasets demonstrate that LLMEraser excels in efficiently managing various unlearning scenarios while maintaining the overall integrity and efficacy of the models.

1 INTRODUCTION

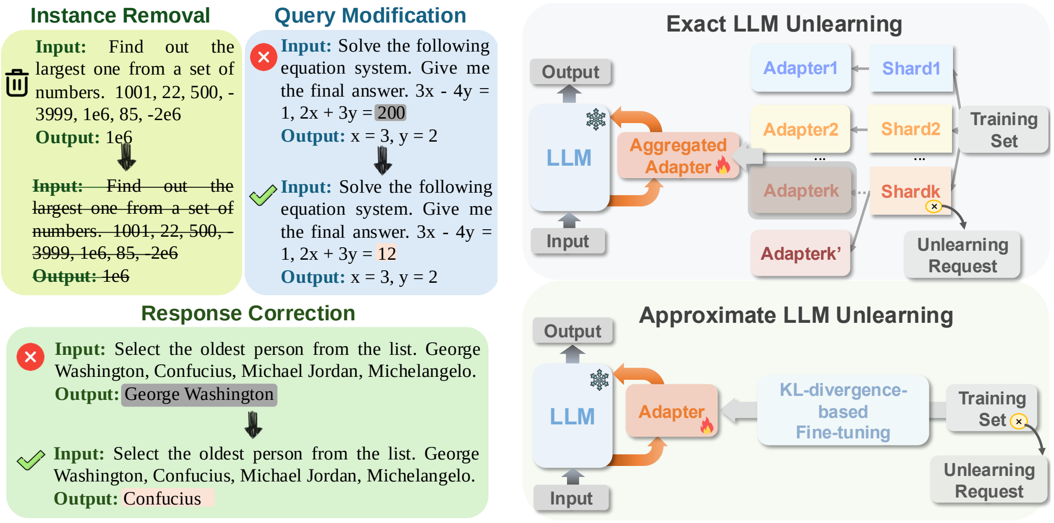
Large language models (LLMs) demonstrate remarkable capabilities in knowledge understanding and complex reasoning (Li et al., 2023; Zhang et al., 2024b; Li, 2024; Li et al., 2024; Lee et al., 2024), having sparked increasing interest in adapting LLMs to specific domains through fine-tuning techniques (Li & Liang, 2021; Dettmers et al., 2023; Zhang et al., 2023; Zaken et al., 2022). Among them, Parameter-Efficient Fine-Tuning (PEFT) (Li & Liang, 2021; Liu et al., 2021), such as LoRA (Hu et al., 2022), has emerged as the mainstream paradigm, offering significant reductions in resource costs by fine-tuning only a small subset of parameters. While highly effective, the reliance on domain-specific data for fine-tuning raises concerns regarding data leakage and privacy (Lu et al., 2024; Blanco-Justicia et al., 2024), such as potentially memorizing or propagating sensitive, biased, copyrighted, or harmful information (Liu et al., 2024c; Qu et al., 2024). In this light, researchers have introduced unlearning techniques (Jang et al., 2023; Kurmanji et al., 2023; Kumar et al., 2023) into LLMs, to "forget" specific data without requiring the time-consuming and resource-intensive process of retraining.

Prior efforts in exploring unlearning in LLMs primarily focus on removing specific concepts (Kassem et al., 2023; Jang et al., 2023). A typical example is the erasure of LLM's ability to recall information related to the Harry Potter series (Eldan & Russinovich, 2023). While these efforts yield valuable insights, they risk inadvertently affecting related concepts, such as other novels with similar titles. In this work, we broaden the scope by investigating instance-wise unlearning tasks, which allow us to target more nuanced aspects of model behavior. To this end, we first present various instance-wise unlearning tasks for LLMs, as illustrated in Figure 1. More case studies can be found in Appendix B. Specifically, consider a training instance $z = (x, y)$ in a supervised fine-tuning dataset, where x represents the query and y is the response. We can categorize the LLMs unlearning tasks at the instance level as follows:

- **Instance Removal (IR).** It removes the sample $z = (x, y)$ from the training set.
- **Query Modification (QM).** It adjusts the input tokens in query x, such as removing specific noisy tokens or correcting certain erroneous tokens.

Table 1: A summary of existing LLM unlearning methods and their application scenarios. E and A are abbreviations for Exact unlearning and Approximate unlearning, respectively.

Related Work	Mode	Method	Preserve Model Architecture	Free from Retrain/Pretrain	IR	QM	RC
Retrain	-	Retrain	✓	✗	✓	✓	✓
SISA (Bourtoule et al., 2021)	E	Retrain Sub-model	✗	✗	✓	✓	✓
FairSISA (Kadhe et al., 2023)	E	Retrain Sub-model	✗	✗	✓	✓	✓
APA (Hu et al., 2024b)	E	Retrain Sub-model	✗	✗	✓	✓	✓
Gradient Ascent	A	Fine-tuning	✗	✓	✓	✗	✗
EUL (Chen & Yang, 2023)	A	Fine-tuning	✗	✓	✓	✗	✗
E2URec (Wang et al., 2024)	A	Fine-tuning	✗	✗	✓	✗	✗
LLMEraser (Ours)	A	Parameter Editing	✓	✓	✓	✓	✓



Instance Removal

Input: Find out the largest one from a set of numbers. 1001, 22, 500, -3999, 1e6, 85, -2e6
Output: 1e6

~~**Input:** Find out the largest one from a set of numbers. 1001, 22, 500, -3999, 1e6, 85, -2e6~~
~~**Output:** 1e6~~

Query Modification

Input: Solve the following equation system. Give me the final answer. 3x - 4y = 1, 2x + 3y = 200
Output: x = 3, y = 2

Input: Solve the following equation system. Give me the final answer. 3x - 4y = 1, 2x + 3y = 12
Output: x = 3, y = 2

Response Correction

Input: Select the oldest person from the list. George Washington, Confucius, Michael Jordan, Michelangelo.
Output: George Washington

Input: Select the oldest person from the list. George Washington, Confucius, Michael Jordan, Michelangelo.
Output: Confucius

(a) Taxonomy of LLM unlearning tasks.

(b) Overview of exact/approximate LLM Unlearning.

Figure 1: 1a: A brief description of the different types of LLM unlearning tasks. 1b: The framework of exact LLM unlearning method, approximate unlearning method.

- **Response Correction (RC).** It corrects the model's response y, including updating outdated answers or rectifying incorrect classification results.

In this work, we focus on unlearning the domain-specific data used solely in PEFT, which requires updating the PEFT adapters (*e.g.,* LoRA). Technically, recent LLM-unlearning efforts can be roughly grouped into two categories. **Exact unlearning** approaches divide data into disjoint shards and retrain adapters (Bourtoule et al., 2021; Hu et al., 2024b). Despite effectiveness, these methods have inherent limitations — inevitably destroying the model's original structure and necessitating the retraining cost. **Approximate unlearning** methods, on the other hand, aim to replicate the performance of the retrained model, often aligning the output of the target data closely with randomness through KL-divergence-based PEFT (Liu et al., 2024a; Qu et al., 2024). Nonetheless, this paradigm primarily focuses on data removal (*e.g.,* IR) and hardly corrects biased or inaccurate data (*e.g.,* QM, RC), as it falls short in guiding the output of the target data towards accurate information, rather than mere randomness. See Table 1 for the summary of current LLMs unlearning methods, with detailed descriptions available in Appendix A. Overall, both approaches struggle to efficiently handle these instance-wise LLM unlearning tasks and are not specifically designed for unlearning within the PEFT framework. It calls for a general LLM unlearning method capable of addressing these various tasks.

In pursuit of parameter-efficient unlearning, we identify the influence function (Koh & Liang, 2017) as a promising tool. At its core is to formulate the parameter changes caused by perturbations in the form of the inverse-Hessian-vector product (Agarwal et al., 2016), where Hessian matrix represents the curvature of the loss function *w.r.t.* model parameters. However, the direct application of the influence function to LLMs presents two significant challenges: the expensive cost of calculating the inverse Hessian-vector product for vast model parameters and the cumulative errors introduced by approximation strategies (*e.g.,* stochastic estimation (Agarwal et al., 2016)). Consequently, the use of influence functions for LLM unlearning remains largely underexplored. To fill this research gap, we propose a unified parameter-efficient unlearning framework, LLMEraser, for various instance-wise unlearning tasks. Specifically, for each type of unlearning task, LLMEraser leverages influence functions to directly calculate the parameter changes in the PEFT adapters and then efficiently update the adapter parameters, thus bypassing the need for time-consuming model retraining or fine-tuning. Furthermore, we reformulate the calculation of the inverse-Hessian-vector product into a finite-sum quadratic programming problem (Nesterov, 2013; Beck & Teboulle, 2009), significantly reducing computational complexity while mitigating the approximation errors from stochastic estimation. LLMEraser has several advantages: model-agnostic, applicable to various instance-wise unlearning tasks, and ensuring fast model updates. We conduct experiments on both LLMs and Multimodal Large Language Models (MLLMs), specifically focusing on LLMs for Recommendation (LLM4Rec) as well as MLLM relation mining tasks to validate the effectiveness of LLMEraser. Our extensive evaluations across these diverse scenarios demonstrate that LLMEraser consistently outperforms the state-of-the-art unlearning methods.

2 PRELIMINARY

This section introduces key concepts underpinning our methodology. We cover instruction tuning to enhance LLMs' understanding of human instructions, followed by PEFT, highlighting LoRA for efficient updates. Lastly, we discuss the influence function, which analyzes parameter changes from data perturbations. These foundations set the stage for the techniques discussed later.

2.1 INSTRUCTION TUNING

Instruction tuning is a key technique that leverages carefully curated datasets of human-annotated instructions and corresponding responses to enhance LLMs' capacity to comprehend and respond to human instructions (Wei et al., 2022; Liu et al., 2023b; Sanh et al., 2022). Given a downstream task dataset $\mathcal{Z} = \{z|z = (x, y)\}$ containing n instances, where x represents a description of the human instruction and y is the corresponding response, LLMs are fine-tuned using the following autoregressive (Brown et al., 2020; Touvron et al., 2023a) objective:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P\left(y_t \mid x, y_{<t}; \Phi\right)\right), \tag{1}$$

where Φ is LLMs' parameters, y_t is the t-th token of y, and $y_{<t}$ represents tokens preceding y_t.

2.2 PARAMETER-EFFICIENT FINE-TUNING

LLMs typically consist of billions of parameters, making full fine-tuning computationally expensive. Parameter-Efficient Fine-Tuning (PEFT) addresses this challenge by updating only a small number of the parameters while still achieving satisfactory performance. Among them, LoRA (Hu et al., 2022) stands out as particularly effective, which freezes the original pretrained parameters while introducing pairs of low-rank-decomposition weight matrices to simulate parameter updates. Formally, the optimization objective for LoRA is expressed as follows:

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P\left(y_t \mid x, y_{<t}; \Phi + \Delta\Phi(\Theta)\right)\right), \tag{2}$$

where Θ is the trainable parameters that is significantly smaller in size compared to Φ.

2.3 INFLUENCE FUNCTION

The influence function was first applied in machine learning by Koh & Liang (2017) to analyze the outputs of black-box models. For the dataset $\mathcal{Z}$, we focus on the following empirical risk minimization (Shalev-Shwartz & Ben-David, 2014; Vapnik, 1998; Bartlett & Mendelson, 2002) problem:

$$\hat{\Theta} \in \arg\min_{\Theta} \left\{ R(\mathcal{Z};\Theta) | R(\mathcal{Z};\Theta) := \frac{1}{n} \sum_{(x,y)\in\mathcal{Z}} \mathcal{L}\left((x,y);\Theta\right) \right\}, \tag{3}$$

where Θ is the trainable model parameter and $\hat{\Theta}$ is the minimizer of Equation 3. $\mathcal{L}(\cdot;\Theta)$ is the loss function, and for Equation 2, it is defined as:

$$\mathcal{L}((x,y);\Theta) = -\sum_{t=1}^{|y|} \log\left(P\left(y_t \mid x, y_{<t}; \Phi + \Delta\Phi(\Theta)\right)\right). \tag{4}$$

When a training example (x,y) is upweighted by an infinitesimal amount ϵ, the perturbed loss for $\hat{\Theta}_{\text{new}}(\epsilon)$ can be expressed as:

$$\hat{\Theta}_{\text{new}}(\epsilon) \in \arg\min_{\Theta} \left\{ \widehat{\mathcal{L}}\left(\mathcal{Z},(x,y),\epsilon;\Theta\right) | \widehat{\mathcal{L}}\left(\mathcal{Z},(x,y),\epsilon;\Theta\right) := R(\mathcal{Z};\Theta) + \epsilon\mathcal{L}\left((x,y);\Theta\right) \right\}. \tag{5}$$

When $\epsilon \approx 0$, the parameter change $\Delta\Theta(\epsilon) = \hat{\Theta}_{\text{new}}(\epsilon) - \hat{\Theta}$ can be approximately calculated by applying a Taylor expansion of Equation 3. Please refer to (Koh & Liang, 2017) for detailed derivation. Specifically, $\Delta\Theta(\epsilon)$ can be written as:

$$\Delta\Theta(\epsilon) \approx -\epsilon H_{\hat{\Theta}}^{-1} \nabla_{\Theta}\mathcal{L}\left((x,y);\hat{\Theta}\right), \tag{6}$$

where $H_{\hat{\Theta}} = \nabla_{\Theta}^2 R(\mathcal{Z};\hat{\Theta})$ is the Hessian matrix, $\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta})$ represents the gradient of $\mathcal{L}$ *w.r.t.* parameters Θ, evaluated at $\hat{\Theta}$.

3 METHOD

In this work, we propose LLMEraser, a framework that updates the PEFT adapter parameters to handle various instance-wise unlearning tasks. As shown in Figure 2, our approach leverages the influence function to directly estimate the parameter changes for various unlearning tasks, circumventing the resource-consuming fine-tuning or retraining procedures. Moreover, we present a novel algorithm to accelerate the computation of the inverse Hessian-vector-product in the influence function, enabling its efficient implementations in LLMs. Finally, we summarize how LLMEraser works.

3.1 TAXONOMY OF LLM UNLEARNING TASKS

We focus on instance-wise unlearning tasks for LLMs, specifically for PEFT that uses domain-specific data. For an instance $z = (x,y)$, where x represents the query and y is the response, we propose a taxonomy of unlearning tasks based on the operation applied to the target instance.

Instance Removal (IR). When a specific instance $z = (x,y)$ is either restricted from use or contains harmful content, it necessitates complete elimination from the training set, along with its associated influence on the model.

Query Modification (QM). This category involves modifying the query x, transforming $z = (x,y)$ into $z' = (x',y)$. It could not only delete outdated or incorrect tokens in the query x, such as noisy interactions from a user's history, but also update erroneous or outdated tokens with correct ones.

Response Correction (RC). Here, the focus is on rectifying the output component y of the instance z. That is, replacing $z = (x,y)$ with $z' = (x,y')$. For binary classification tasks, such as answering "Yes" or "No", it corrects mislabeled outputs by flipping the labels. For other tasks, such as multi-class classification or question answering, it is applied to rectify inaccurate responses.

Our proposed taxonomy expands the concept of LLM unlearning beyond the removal of entire instances. It introduces a more fine-grained categorization defined at the token level within both queries and responses, allowing for nuanced control of model behavior.

3.2 LLMERASER

The key strength of LLMEraser lies in its capacity to directly estimate the adapter's parameter changes caused by various unlearning tasks. For the sake of clarity and without sacrificing generality, we employ the loss function in LoRA (*cf.* Equation 4) as our example, while other alternatives would yield similar formulations.

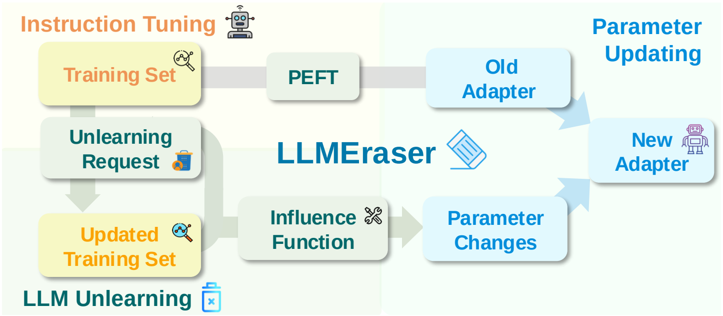


Figure 2: The framework of LLMEraser. The old adapter is obtained through PEFT on domain-specific data. When an unlearning request arrives (e.g., deleting or correcting certain data from the training set), LLMEraser utilizes influence functions to compute the parameter changes caused by such request. These estimated parameter modifications are added to the old adapter's weights, resulting in the new adapter parameters—essentially the unlearned model parameters.

To develop a unified approach for solving all unlearning tasks in our taxonomy, we begin by considering a general case where perturbations are applied to both the query (x) and response (y) components of an instance z. This generalized framework allows us to model each specific unlearning task as a special case of this perturbation scenario. Formally, we define the perturbation δ applied to z as $z_\delta = (x + \delta_x, y + \delta_y)$, where δ_x and δ_y represent perturbations to the query and response, respectively. We now formulate the perturbed empirical risk minimization problem as:

$$\widehat{\Theta_\delta}(\epsilon) \in \arg\min_{\Theta} \left\{ R(\mathcal{Z};\Theta) + \epsilon\mathcal{L}\left((x+\delta_x, y+\delta_y);\Theta\right) - \epsilon\mathcal{L}\left((x,y);\Theta\right) \right\}, \tag{7}$$

where $\widehat{\Theta_\delta}(\epsilon)$ is the minimizer of the optimization problem after applying a perturbation δ of magnitude ϵ to the sample z. Following the derivation in (Koh & Liang, 2017), when the sample size n is sufficiently large, by taking $\epsilon = \frac{1}{n}$ (*i.e.,* $\epsilon \approx 0$), we can safely estimate the parameter change $\Delta\Theta_\delta$ as follows:

$$\Delta\Theta_\delta \approx \frac{1}{n}\left(\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\right)^{-1} \left(\mathcal{G}(x,y) - \mathcal{G}(x+\delta_x, y+\delta_y)\right), \tag{8}$$

where $\mathcal{G}(x,y)$ is an abbreviations for $\nabla_\Theta\mathcal{L}\left((x,y);\hat{\Theta}\right)$. Next, we present the perturbations and corresponding parameter changes for different unlearning tasks.

- **Instance Removal.** The deletion of data corresponds to the perturbation function in Equation 5. By setting $\epsilon = -\frac{1}{n}$ like Equation 6, it is equivalent to removing instance z. The set of deleted instances is denoted as $\mathcal{S}_{\text{IR}}$. By aggregating the gradients of all deleted instances, the parameter change $\Delta\Theta_{\text{IR}}$ can be expressed as follows:

$$\Delta\Theta_{\text{IR}} \approx \frac{1}{n}\left(\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\right)^{-1} \sum_{(x,y)\in\mathcal{S}_{\text{IR}}} \mathcal{G}(x,y). \tag{9}$$

- **Query Modification**. Modifying certain tokens in the query x is equivalent to perturbing x with δ_x, where δ_x represents deleting noisy tokens or correcting inaccurate tokens, while keeping the response unchanged (*i.e.,* $\delta_y = 0$). Hence, the perturbed instance z is represented as $z_\delta = (x+\delta_x, y)$, with the set of instances requiring the removal or modification of specific tokens represented by $\mathcal{S}_{\text{QM}}$. By aggregating the gradients of all instances in $\mathcal{S}_{\text{QM}}$, the parameter change $\Delta\Theta_{\text{QM}}$ induced by query modification can be shown as follows:

$$\Delta\Theta_{\text{QM}} \approx \frac{1}{n}\left(\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\right)^{-1} \left(\sum_{(x,y)\in\mathcal{S}_{\text{QM}}} \mathcal{G}(x,y) - \sum_{(x+\delta_x,y)\in\mathcal{S}_{\text{QM}}} \nabla_\Theta\mathcal{G}(x+\delta_x,y) \right). \tag{10}$$

- **Response Correction.** Correcting the response solely corresponds to $\delta_x = 0$ while perturbing the response y with δ_y. Here δ_y represents updates to outdated answers or adjustments to erroneous classification results. With $z_\delta = (x, y+\delta_y)$, the set of instances with rectified labels is $\mathcal{S}_{\text{RC}}$. The parameter change $\Delta\Theta_{\text{RC}}$ is as follows:

$$\Delta\Theta_{\text{RC}} \approx \frac{1}{n}\left(\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\right)^{-1} \left(\sum_{(x,y)\in\mathcal{S}_{\text{RC}}} \mathcal{G}(x,y) - \sum_{(x,y+\delta_y)\in\mathcal{S}_{\text{RC}}} \mathcal{G}(x,y+\delta_y) \right). \tag{11}$$

However, computing inverse Hessian-vector-product results presents significant challenges. Although CG (Hestenes et al., 1952; Fletcher, 2000; Shewchuk et al., 1994) shows some promise, it requires full-batch gradient computation (Koh & Liang, 2017), making it impractical for large-scale datasets. Stochastic estimation (Agarwal et al., 2016) expands $(\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta}))^{-1}$ into a truncated power series and iteratively estimates parameter changes, but it suffers from cumulative approximation errors (Blanco-Justicia et al., 2024; Basu et al., 2021). Next, we elaborate a new efficient and scalable algorithm for computing $\Delta\Theta_{\text{Task}}$ for different unlearning tasks.

3.3 A NEW ALGORITHM FOR COMPUTING PARAMETER CHANGES

LLMEraser reformulates the calculation of parameter changes as solving an equivalent optimization problem expressed in summation form, enabling efficient resolution using mini-batch algorithms. Specifically, we focus on the following optimization problem regarding Δ:

$$\min_{\Delta} F(\Delta) := \frac{1}{2}\Delta^\top \nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\Delta - \langle b, \Delta\rangle, \tag{12}$$

where $\langle , \rangle$ represents the inner product of vectors, and b is defined as:

$$b = \begin{cases} \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{IR}}} \mathcal{G}(x,y), & \text{if Task = IR} \\ \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{IM}}} \mathcal{G}(x,y) - \frac{1}{n}\sum_{(x+\delta_x,y)\in\mathcal{S}_{\text{IM}}} \mathcal{G}(x+\delta_x,y), & \text{if Task = IM} \\ \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{RC}}} \mathcal{G}(x,y) - \frac{1}{n}\sum_{(x,y+\delta_y)\in\mathcal{S}_{\text{RC}}} \mathcal{G}(x,y+\delta_y), & \text{if Task = RC} \end{cases}. \tag{13}$$

Since $\hat{\Theta}$ is the minimizer of Equation 3, it satisfies the second-order necessary optimality condition (Nocedal & Wright, 1999; Luenberger et al., 1984; Bertsekas, 1997), resulting in the matrix $\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})$ being symmetric and positive semidefinite. Thus, Equation 12 is essentially a convex quadratic problem, with a gradient of $\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\Delta - b$. Given that $\Delta\Theta_{\text{Task}}$ can be interpreted as the solution to the linear system $\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})\Delta = b$, addressing $\Delta\Theta_{\text{Task}}$ is effectively equivalent to optimizing Equation 12. Due to the summation form of $\nabla^2_\Theta R(\mathcal{Z};\hat{\Theta})$, Equation 12 can be reformulated as the following finite-sum formation:

$$F(\Delta) = \frac{1}{n}\sum_{(x,y)\in\mathcal{Z}} f\left((x,y),\Delta\right), \tag{14}$$

where $f((x,y),\Delta)$ is defined as:

$$f((x,y),\Delta) = \frac{1}{2}\Delta^\top \nabla^2_\Theta \mathcal{L}\left((x,y),\hat{\Theta}\right)\Delta + \langle b, \Delta\rangle. \tag{15}$$

By employing scalable algorithms (*e.g.,* SGD) to optimize problem 12, we can obtain the solution for $\Delta\Theta_{\text{Task}}$. It is worth noting that both the function value and the gradient can be efficiently computed using the Hessian-vector-product (HVP)[1], reducing the complexity from $\mathcal{O}(p^2)$ to $\mathcal{O}(p)$ (Pearlmutter, 1994), where p is the number of trainable parameters. The pseudocode for computing parameter changes can be found in Appendix C. Error analysis for our proposed algorithm can be found in Appendix E.

3.4 THE WORKFLOW OF LLMERASER

LLMEraser focuses on unlearning domain-specific data and updating the parameters of the PEFT adapters. Overall, the workflow of LLMEraser is as follows:

- Use domain-specific data with PEFT to obtain the old adapter.
- Receive the unlearning request.
- LLMEraser utilizes influence functions to compute the changes in model parameters caused by the unlearning request.
- Add the computed parameter changes to the old adapter's parameters to obtain the unlearned model parameters.

---

[1] HVP has a corresponding implementation in PyTorch; refer to https://pytorch.org/docs/stable/autograd.html for details.

Table 2: Experimental results on the instance removal task with 5% of training data removed, using TALLRec as the LLM4Rec model on the BookCrossing dataset.

	Original	Retrain	Gradient Ascent	E2URec	**LLMEraser (Ours)**
AUC	0.6400	0.6357	0.6187	0.6205	**0.6319**

4 EXPERIMENT

In this section, we carry out extensive experiments to assess the performance and efficiency of LLMEraser. The experiments are designed to explore the following key research questions: **RQ1:** How does LLMEraser perform across various unlearning tasks? **RQ2:** How does LLMEraser perform at different unlearning ratios? **RQ3:** How does the efficiency of LLMeraser compared to other unlearning methods?

4.1 EXPERIMENTAL SETUPS

We conduct experiments on both LLMs and Multimodal Large Language Models (MLLMs), focusing specifically on LLMs for Recommendation (LLM4Rec) (Bao et al., 2023; Liao et al., 2024) and MLLM relation mining tasks (Wu et al., 2024; Ye et al., 2024), to validate the effectiveness of our proposed LLMEraser. We choose LLaMA2-7B (Touvron et al., 2023b) as our backbone LLM and LLaVA 1.5-7B (Liu et al., 2023a) for the MLLM experiments. Comprehensive details on task, datasets, baselines, and evaluation metrics for our proposed LLMEraser can be found in Appendix D.1.

4.2 RESULTS ANALYSIS FOR VARIOUS UNLEARNING TASKS (RQ1)

We design a variety of comprehensive experiments to thoroughly validate the effectiveness of LLMEraser across the three unlearning tasks we have proposed. More experimental results on the generative task can be found in Appendix G.

4.2.1 RESULTS ANALYSIS ON INSTANCE REMOVAL

For instance removal, we directly delete a proportion of training instances and subsequently evaluate the performance of each unlearning method. The experimental results on LLM4Rec are shown in Table 2. We can find that: (1) LLMEraser closely mirrors the performance of Retrain. The performance gap between LLMEraser and Retrain is merely 0.0038, constituting only 0.6% of Retrain's performance. This can be attributed to our method's direct estimation of the parameter changes between the retrained model and the original model, allowing for a highly accurate calculation of these changes. (2) Other unlearning methods exhibit notable declines in model performance. Specifically, Gradient Ascent and E2URec show average decreases of 2.7% and 2.4%, respectively, as they do not explicitly aim to approximate the Retrain model during the fine-tuning process.

4.2.2 RESULTS ANALYSIS ON QUERY MODIFICATION & RESPONSE CORRECTION

Adversarial attack experiments are widely employed to assess the efficacy of data modification for unlearning techniques (Wu et al., 2023; Moon et al., 2024; Cha et al., 2024). The core idea is first randomly introducing corrupted instances into the dataset, which inevitably leads to a decline in model performance, and then leveraging unlearning techniques to correct these noisy data on the model. Following this setting, we evaluate the performance of LLMEraser in both query modification and response correction tasks.

For query modification, we conduct experiments on the LLM4Rec task by adding adversarial noise to the user interaction sequences, *i.e.,* randomly deleting some items from the sequences (Interaction Removal) or replacing them with corrupt ones (Interaction Replacement), and then using LLMEraser to rectify the data. Table 3 presents the experimental results. We can observe that: (1) LLMEraser brings a substantial utility gain to the model compared to the corrupted baseline, significantly reducing the negative impact of noisy data. Specifically, it achieves an average improvement of 5.1% compared to the corrupted model in both settings, with a peak increase of 5.5% in interaction removal setting. Moreover, its performance is closest to that of Retrain, demonstrating its

Table 3: Experimental results on the QM task, using LLaRA as the LLM4Rec model on the MovieLens and LastFM datasets. "10% Interaction Removal" refers to 10% of users have items removed from their interaction sequences, "5% Interaction Replacement" refers to 5% of users have items replaced with noisy interactions. **Corrupted** refers to the model trained with the noisy data.

	Method	Movielens		LastFM	
		HitRatio@1	ValidRatio	HitRatio@1	ValidRatio
10% Interaction Removal	Retrain	0.4565	0.9684	0.4508	1.0000
	Corrupted	0.4222	0.9375	0.4344	1.0000
	SISA	0.4130	0.9684	0.4132	0.9918
	RecEraser	0.2717	0.9684	0.4298	0.9918
	LLMEraser (Ours)	**0.4456**	**0.9684**	**0.4463**	0.9918
5% Interaction Replacement	Retrain	0.4565	0.9684	0.4508	1.0000
	Corrupted	0.4316	0.9684	0.4344	0.9918
	SISA	0.3804	0.9684	0.4050	0.9918
	RecEraser	0.3152	0.9684	0.3689	1.0000
	LLMEraser (Ours)	**0.4516**	**0.9789**	**0.4426**	**1.0000**

Table 4: Experimental results on the MM-SPUBENCH for RC tasks, where **Corrupted** denotes we assign wrong labels for 40% of the training samples.

Method	MM-SPUBENCH									Average	All
	BG	TN	CO	RS	Col.	Ori.	LS	PA	Sha.		
Retrain	0.88	0.80	0.83	1.00	0.78	0.86	0.86	0.66	0.70	0.82	0.84
Corrupted	0.76	0.62	0.67	0.80	0.67	0.76	0.65	0.68	0.67	0.70	0.71
SISA	0.84	0.65	0.79	1.00	0.64	0.79	0.86	0.73	0.57	0.76	0.77
LLMEraser	**0.86**	**0.70**	**0.80**	**1.00**	**0.78**	**0.85**	**0.84**	**0.76**	**0.67**	**0.81**	**0.81**

effectiveness in correcting inaccurate input information. (2) SISA and RecEraser fail to improve performance. Their average results in both settings decreased by 7.0% and 31.3% compared to the corrupted baseline. The reasons may lie in their dataset partitioning and submodel retraining strategy, potentially leading to a loss of crucial contextual information and introducing inconsistencies in learned representations. (3) RecEraser underperforms SISA in most cases. Designed on traditional recommendation models, RecEraser relies on users' collaborative signals to optimize shard partitioning; however, this strategy fails to effectively adapt to LLM4Rec.

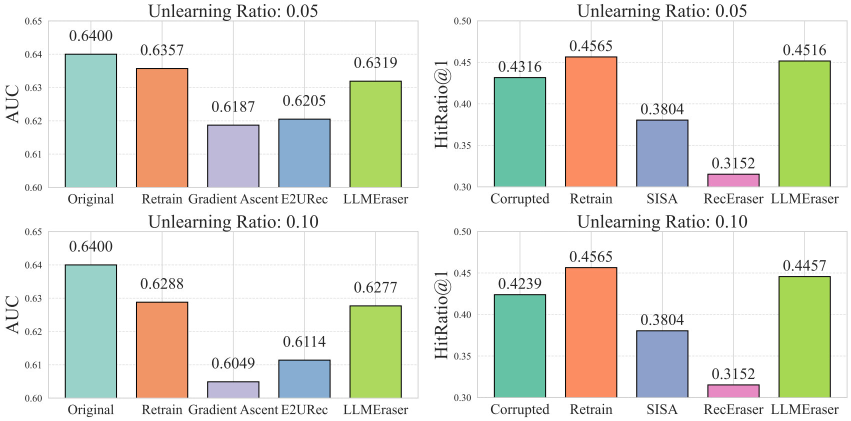
For response correction, we introduce noise into the training data of the MLLMs task by randomly assigning incorrect labels to a portion of the samples. In the spurious biases task for MLLMs, we reverse 40% the original "yes/no" labels. For the hard hallucination mining task in MLLMs, we assign random labels to 40% of the samples. We leverage LLM unlearning to mitigate the negative impact of such noisy data, aiming to approximate the performance of retraining with clean data. The experimental results of response correction unlearning task on spurious biases task and hard hallucination mining task are presented in Table 4 and 5, respectively. We can draw the following observations: (1) LLMEraser effectively performs response correction, achieving average improvements of 14.2% and 18.9% on the spurious biases task and hard hallucination mining task, respectively, compared to the corrupted baseline. Compared to other methods, LLMEraser shows the smallest performance gap relative to Retrain. On the spurious biases task and hard hallucination mining task, the average differences with Retrain are 0.024 and 0.048, which account for 2.9% and 7.5% of Retrain's performance, respectively. Whether addressing label reversal in binary classification or correcting labels in multi-class scenarios, LLMEraser can eliminate the negative impact of noisy labels and restore them to their clean, original state. (2) The improvement brought by SISA is not significant. Although SISA ensures that dirty data is replaced with clean data during retraining, its data segmentation strategy can inevitably hurt model performance.

4.3 RESULTS ANALYSIS FOR DIFFERENT UNLEARNING RATIOS (RQ2)

To assess the sensitivity of various unlearning methods to different scales of unlearning data, we conduct experiments using different unlearning ratios in instance removal and query modification tasks.

Table 5: Experimental results on the R-BENCH for RC tasks, where **Corrupted** denotes we assign wrong labels for 40% of training samples.

Method	Recall	F1-Score	Precision	Accuracy	Yes
Retrain	0.70	0.66	0.63	0.65	0.55
Corrupted	0.47	0.50	0.53	0.54	0.44
SISA	0.47	0.49	0.52	0.52	0.45
LLMEraser (Ours)	**0.68**	**0.63**	**0.58**	**0.56**	**0.50**



(a) Impact of unlearning ratio in IR. (b) Impact of unlearning ratio in QM.

Figure 3: 3a: Experimental results of the instance removal task using TallRec as the LLM4Rec model on the BookCrossing dataset, where 5% and 10% of the training data were randomly deleted. 3b: Experimental results of the query modification task using LLaRA as the LLM4Rec model on the MovieLens dataset, where interactions were randomly removed from 5% and 10% of users.

For the instance removal, we employ TallRec as the LLM4Rec framework, where 5% and 10% of instances are removed. Meanwhile, for query modification, LLARA is utilized as the backbone, where 5% and 10% of user interactions are deleted. The experimental results are shown in Figure 3. From these results, we can find that: (1) In the instance removal task, LLMEraser consistently performs closest to Retrain across different unlearning ratio settings, with an average performance decline of only 1.18%. This indicates that LLMEraser can effectively delete data while minimizing the negative impact on model performance. (2) In the query modification task, LLMEraser consistently achieves the best performance across various unlearning ratios, with an average improvement of 4.9% compared to corrupted method. Notably, at an unlearning ratio of 10%, the relative improvement reaches 5.1%. The average difference between LLMEraser and Retrain is only 0.0079. In comparison to SISA and RecEraser, LLMEraser demonstrates a superior ability to maintain model utility. This highlights the effectiveness of LLMEraser, demonstrating its robust performance across varying unlearning demands. (3) We observe an interesting phenomenon in query modification task under adversarial attack settings, with a sufficiently high unlearning ratio (in this case, 5% and 10%), both SISA and Receraser require retraining all shards with the same clean data, resulting in equivalent outcomes. Despite the direct use of clean data for retraining, they still struggle to obtain optimal model performance.

4.4 RESULTS ANALYSIS FOR UNLEARNING EFFICIENCY (RQ3)

Efficiency is a key metric in evaluating unlearning techniques, particularly for LLMs. We here conduct experiments, comparing our proposed LLMEraser against existing techniques. For a fair comparison, we report the execution time in the QM task, where 5% of users have items replaced with noisy interactions. All methods are run on a single Nvidia A100 GPU. Table 6 presents the results. We can observe that: (1) Due to the parallel training of sub-models, the retraining time of both SISA and RecEraser can be reduced to some extent. However, RecEraser requires data partitioning based on similarity, which introduces additional computational overhead.

Table 6: Execution time in the QM task.

Method	Time (s)
Retrain	5.4×10^4
SISA	1.8×10^4
RecEraser	2.0×10^4
LLMEraser	1.4×10^3

Moreover, both methods remain highly inefficient as unlearning requests necessitate retraining of the adapters. (2) In contrast, our proposed LLMEraser exhibits remarkable efficiency in handling unlearning tasks. By directly modifying model parameters, LLMEraser achieves a speedup of approximately 31.25 times compared to retraining, requiring only about 1.4×10^3seconds to update the parameters. This reduction in execution time demonstrates the effectiveness of our approach in accelerating the computation of parameter changes. Additional experimental results and related analyses on the memory usage and execution time of LLMEraser can be found in Appendix F.

5 LIMITATIONS

LLMEraser offers efficient parameter updates without the need for retraining, making it versatile across different unlearning tasks while also reducing computational overhead. Despite the improvements brought by LLMEraser, its potential shortcomings should not be overlooked. Calculating parameter changes for different unlearning tasks requires accessing the gradient information of the target data and assumes the availability of the training set. Furthermore, the influence function's reliance on the first-order Taylor expansion of the optimization objective leads to inevitable estimation errors, representing an inherent limitation of such an approach.

6 CONCLUSION AND FUTURE WORK

This paper introduces LLMEraser, a unified parameter-efficient unlearning framework. By systematically categorizing and addressing various unlearning tasks, LLMEraser leverages influence functions for parameter adjustments, circumventing the cumbersome retraining processes common in traditional methods. Extensive experiments on benchmark datasets show that LLMEraser excels in efficiently handling various unlearning tasks while preserving the overall integrity and efficacy of the models. Additionally, LLMEraser opens new avenues for future research, encouraging the exploration of enhanced unlearning techniques and their implications in diverse applications, such as data privacy and ethical AI. Future studies could explore the broader applicability of LLMEraser and potential optimizations for its computational efficiency and accuracy.

ETHICS STATEMENT

This work is primarily foundational in instance-wise unlearning for LLMs, focusing on the development of a more efficient parameter-efficient unlearning framework. Its primary aim is to contribute to the academic community by enhancing the understanding and implementation of LLM applications in specific domains. We do not foresee any direct, immediate, or negative societal impacts stemming from the outcomes of our research.

REPRODUCIBILITY STATEMENT

All results presented in this work are fully reproducible. We provide the code for our method and baseline models in the supplementary materials. The optimal hyperparameters utilized in our experiments are detailed in Appendix D.2. Our code is available at `https://anonymous.4open.science/r/LLMEraser-0376`.

REFERENCES

Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *CoRR*, abs/1602.03943, 2016.

Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. Chainforge: A visual toolkit for prompt engineering and LLM hypothesis testing. In *CHI*, pp. 304:1–304:18. ACM, 2024.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, pp. 1007–1014. ACM, 2023.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002.

Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *ICLR*. OpenReview.net, 2021.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Alberto Blanco-Justicia, Najeeb Jebreel, Benet Manzanares-Salor, David Sánchez, Josep Domingo-Ferrer, Guillem Collell, and Kuan Eeik Tan. Digital forgetting in large language models: A survey of unlearning methods. *CoRR*, abs/2404.02062, 2024.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *SP*, pp. 141–159. IEEE, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

William Cain. Prompting change: exploring prompt engineering in large language model ai and its potential to transform education. *TechTrends*, 68(1):47–57, 2024.

Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik (eds.). *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '11, Chicago, Illinois, USA, October 27, 2011*, 2011. ACM.

Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In *AAAI*, pp. 11186–11194. AAAI Press, 2024.

Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *WWW*, pp. 2768–2777. ACM, 2022.

Jiaao Chen and Diyi Yang. Unlearn what you want to forget: Efficient unlearning for llms. In *EMNLP*, pp. 12041–12052. Association for Computational Linguistics, 2023.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. Dialogsum: A real-life scenario dialogue summarization dataset. In *ACL/IJCNLP (Findings)*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 5062–5074. Association for Computational Linguistics, 2021.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor.*, 25(2):42–61, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *NeurIPS*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

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

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016.

Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019.

Cunchen Hu, Heyang Huang, Liangliang Xu, Xusheng Chen, Jiang Xu, Shuang Chen, Hao Feng, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, and Yizhou Shan. Inference without interference: Disaggregate LLM inference for mixed downstream workloads. *CoRR*, abs/2401.11181, 2024a.

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. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiyu Hu, Yang Zhang, Minghao Xiao, Wenjie Wang, Fuli Feng, and Xiangnan He. Exact and efficient unlearning for large language model-based recommendation. *CoRR*, abs/2404.10327, 2024b.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In *ACL (1)*, pp. 14389–14408. Association for Computational Linguistics, 2023.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM maybe longlm: Self-extend LLM context window without tuning. *CoRR*, abs/2401.01325, 2024.

Swanand Ravindra Kadhe, Anisa Halimi, Ambrish Rawat, and Nathalie Baracaldo. Fairsisa: Ensemble post-processing to improve fairness of unlearning in llms. *CoRR*, abs/2312.07420, 2023.

Aly M. Kassem, Omar Mahmoud, and Sherif Saad. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *EMNLP*, pp. 4360–4379. Association for Computational Linguistics, 2023.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017.

Vinayshekhar Bannihatti Kumar, Rashmi Gangadharaiah, and Dan Roth. Privacy adhering machine un-learning in NLP. In *IJCNLP (Findings)*, pp. 268–277. Association for Computational Linguistics, 2023.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *NeurIPS*, 2023.

Chanhee Kwak, Junyeong Lee, Kyuhong Park, and Heeseok Lee. Let machines unlearn - machine unlearning and the right to be forgotten. In *AMCIS*. Association for Information Systems, 2017.

Byung-Kwan Lee, Beomchan Park, Chae Won Kim, and Yong Man Ro. Collavo: Crayon large language and vision model. In *ACL (Findings)*, pp. 1121–1138. Association for Computational Linguistics, 2024.

Likun Li, Haoqi Zeng, Changpeng Yang, Haozhe Jia, and Di Xu. Block-wise lora: Revisiting fine-grained lora for effective personalization and stylization in text-to-image generation. *CoRR*, abs/2403.07500, 2024.

Shaoxu Li. Diffstyler: Diffusion-based localized image style transfer. *CoRR*, abs/2403.18461, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP (1)*, pp. 4582–4597. Association for Computational Linguistics, 2021.

Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu Lee Wang, Qing Li, and Xiaoqin Zhong. Label supervised llama finetuning. *CoRR*, abs/2310.01208, 2023.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *SIGIR*, pp. 1785–1795. ACM, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *CoRR*, abs/2310.03744, 2023a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35, 2023b.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. Rethinking machine unlearning for large language models. *CoRR*, abs/2402.08787, 2024a.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602, 2021.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. Understanding llms: A comprehensive overview from training to inference. *CoRR*, abs/2401.02038, 2024b.

Zheyuan Liu, Guangyao Dou, Zhaoxuan Tan, Yijun Tian, and Meng Jiang. Towards safer large language models through machine unlearning. In *ACL (Findings)*, pp. 1817–1829. Association for Computational Linguistics, 2024c.

Weikai Lu, Ziqian Zeng, Jianwei Wang, Zhengdong Lu, Zelin Chen, Huiping Zhuang, and Cen Chen. Eraser: Jailbreaking defense in large language models via unlearning harmful knowledge. *CoRR*, abs/2404.05880, 2024.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. QUARK: controllable text generation with reinforced unlearning. In *NeurIPS*, 2022.

David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

Saemi Moon, Seunghyuk Cho, and Dongwoo Kim. Feature unlearning for pre-trained gans and vaes. In *AAAI*, pp. 21420–21428. AAAI Press, 2024.

Daye Nam, Andrew Macvean, Vincent J. Hellendoorn, Bogdan Vasilescu, and Brad A. Myers. Using an LLM to help with code understanding. In *ICSE*, pp. 97:1–97:13. ACM, 2024.

Yurii E. Nesterov. Gradient methods for minimizing composite functions. *Math. Program.*, 140(1): 125–161, 2013.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: Language models as few shot unlearners. *arXiv preprint arXiv:2310.07579*, 2023.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160, 1994.

Youyang Qu, Ming Ding, Nan Sun, Kanchana Thilakarathna, Tianqing Zhu, and Dusit Niyato. The frontier of data erasure: Machine unlearning for large language models. *CoRR*, abs/2403.15779, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *ICLR*. OpenReview.net, 2022.

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

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

Wenbo Shang and Xin Huang. A survey of large language models on generative graph analytics: Query, learning, and applications. *CoRR*, abs/2404.14809, 2024.

Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain. 1994.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *SIGIR*, pp. 491–500. ACM, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b.

Vladimir Vapnik. Statistical learning theory. *John Wiley & Sons google schola*, 2:831–842, 1998.

Hangyu Wang, Jianghao Lin, Bo Chen, Yang Yang, Ruiming Tang, Weinan Zhang, and Yong Yu. Towards efficient and effective unlearning of large language models for recommendation. *CoRR*, abs/2403.03536, 2024.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*. OpenReview.net, 2022.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. GIF: A general graph unlearning strategy via influence function. In *WWW*, pp. 651–661. ACM, 2023.

Mingrui Wu, Jiayi Ji, Oucheng Huang, Jiale Li, Yuhang Wu, Xiaoshuai Sun, and Rongrong Ji. Evaluating and analyzing relationship hallucinations in large vision-language models. In *ICML*. OpenReview.net, 2024.

Junjie Xu, Zongyu Wu, Minhua Lin, Xiang Zhang, and Suhang Wang. LLM and GNN are complementary: Distilling LLM for multimodal graph learning. *CoRR*, abs/2406.01032, 2024a.

Xuhai Xu, Bingsheng Yao, Yuanzhe Dong, Saadia Gabriel, Hong Yu, James A. Hendler, Marzyeh Ghassemi, Anind K. Dey, and Dakuo Wang. Mental-llm: Leveraging large language models for mental health prediction via online text data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(1):31:1–31:32, 2024b.

Wenqian Ye, Guangtao Zheng, Yunsheng Ma, Xu Cao, Bolin Lai, James M. Rehg, and Aidong Zhang. Mm-spubench: Towards better understanding of spurious biases in multimodal llms. *CoRR*, abs/2406.17126, 2024.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. Unlearning bias in language models by partitioning gradients. In *ACL (Findings)*, pp. 6032–6048. Association for Computational Linguistics, 2023.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *ICML*. OpenReview.net, 2024.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL (2)*, pp. 1–9. Association for Computational Linguistics, 2022.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *ICLR*. OpenReview.net, 2024a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*. OpenReview.net, 2023.

You Zhang, Jin Wang, Liang-Chih Yu, Dan Xu, and Xuejie Zhang. Personalized lora for human-centered text understanding. In *AAAI*, pp. 19588–19596. AAAI Press, 2024b.

Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge without any modification on update rules. In *NeurIPS*, 2022.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW*, pp. 22–32. ACM, 2005.

A OVERVIEW OF EXISTING LLM UNLEARNING

- **SISA (Bourtoule et al., 2021):** It works by dividing the training dataset into partitions, allowing for targeted unlearning of specific instances. The methodology typically involves the following steps: data partitioning, retraining, and aggregation. However, a notable limitation of SISA is that it does not preserve the model architecture and requires retraining of sub-models, which can lead to increased computational costs.
- **FairSISA (Kadhe et al., 2023):** FairSISA improves upon SISA by incorporating fairness enhancements. It still relies on the paradigm of retraining sub-models to handle unlearning requests. This approach inherently alters the model architecture and necessitates the retraining of the sub-models, which can limit the flexibility and efficiency of the unlearning process.
- **APA (Hu et al., 2024b):** This study introduces the first exact unlearning approach for large language model-based recommendation (LLMRec), focusing on the removal of personal data to comply with privacy regulations. The Adapter Partition and Aggregation (APA) method is proposed, which combines data partitioning with parameter aggregation to reduce inference latency while maintaining performance. This approach enables efficient unlearning without incurring the extra costs typically associated with traditional methods. However,it can affect the integrity of the adapter structure and necessitates retraining of sub-models.
- **Gradient Ascent:** It utilizes the gradient of the target instance to fine-tune the adapter by moving in the direction of the negative gradient of the deleted data. However, this approach is not effective for input modification and output correction tasks, as gradient ascent of target instances cannot adequately handle these scenarios.
- **EUL (Chen & Yang, 2023):** This work introduces a lightweight approach for LLMs to efficiently forget specific information without complete retraining. It incorporates unlearning layers into transformer architectures, utilizing a selective teacher-student formulation, and employs a fusion mechanism to combine multiple unlearning layers into a unified layer. This enables LLMs to dynamically handle a sequence of deletion requests while maintaining model performance. The introduction of adapters alters the model's structure, and the KL divergence-based methods are only effective for instance removal tasks, as obtaining a model trained on clean data is not feasible.
- **E2URec (Chen & Yang, 2023):** This method uses lightweight LoRA modules and a teacher-student framework to forget specific data while maintaining performance. However, the extra LoRA module changes the original model architecture, and the teacher-student framework requires pretraining on both retained and forgotten data, which is intricate and cannot perform well on other tasks like editing.

B MORE EXAMPLES OF VARIOUS UNLEARNING TASKS

Instance Removal Case Study

Query ~~Is CLM a stock or an ETF?~~

Response ~~Cornerstone Strategic Value Fund, Inc. is a diversified, closed-end management investment company. It was incorporated in Maryland on May 1, 1987 and commenced investment operations on June 30, 1987. The Fund's shares of Common Stock are traded on the NYSE MKT under the ticker symbol "CLM."[1] That essentially means that CLM is a company all of whose assets are held as tradable financial instruments OR EQUIVALENTLY CLM is an ETF that was created as a company in its own right. That it was founded in the 80s, before the modern definition of ETFs really existed, it is probably more helpful to think of it by the first definition as the website mentions that it is traded as common stock so its stock holds more in common with stock than ETFs. [1] http://www.cornerstonestrategicvaluefund.com/~~

Figure 4: Instance Removal Case Study.

Instance Removal Case Study

Query	~~Given the user's historical interactions, please determine whether the user will enjoy the target new movie by answering "Yes" or "No".~~ ~~User's liked items: GodFather.~~ ~~User's disliked items: Star Wars.~~ ~~Target new movie: Iron Man~~
Response	~~No.~~

Query Modification Case Study

Query	This user has watched: The Rich Man's Wife [emb], Air Force One [emb], Murder at 1600 [emb], Absolute Power in the previous [emb]. Please predict the next movie this user will watch. Choose the answer from the following 10 movie titles: Face/Off [emb], Primal Fear [emb], Ransom [emb], Men in Black [emb], Twelve Monkeys [emb], Lone Star [emb], Mr. Holland's Opus [emb], Jackie Chan's First Strike [emb], Waiting for Guffman [emb], The Long Kiss Goodnight [emb]. Answer:
Response	Face/Off

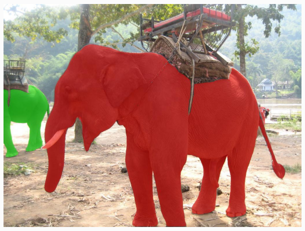
After Query Modification

Query	This user has watched: ~~The Rich Man's Wife [emb],~~ Air Force One [emb], Murder at 1600 [emb], Absolute Power in the previous [emb]. Please predict the next movie this user will watch. Choose the answer from the following 10 movie titles: Face/Off [emb], Primal Fear [emb], Ransom [emb], Men in Black [emb], Twelve Monkeys [emb], Lone Star [emb], Mr. Holland's Opus [emb], Jackie Chan's First Strike [emb], Waiting for Guffman [emb], The Long Kiss Goodnight [emb]. Answer:

Figure 5: Instance Removal Case Study & Query Modification Case Study.

Response Correction Case Study

Query Is the elephant in red mask standing next to a tree in green mask?



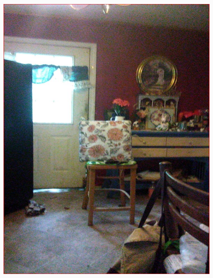
Response Yes

After Response Correction

Response No

Response Correction Case Study

Query You are a helpful assistant that can answer questions for an image. I will provide you 4 options.\nResponse Format\nChoice: A single character from A, B, C, D.\nWhich feature best indicates the identity of the object that has a floral pattern and is placed on a chair?\nChoices:A. The object's soft texture\nB. The indoor setting\nC. The wooden chair\nD. The background clutter



Response D

After Response Correction

Response A

Figure 6: Response Correction Case Study.

C ALGORITHM FOR CALCULATING PARAMETER CHANGES

The algorithm for calculating the parameter changes $\Delta\Theta_{\text{Task}}$ can be found in Algorithm 1. This algorithm accelerates the computation of parameter changes resulting from unlearning requests and is applicable in large-scale data scenarios.

Algorithm 1 Calculate Parameter Changes $\Delta\Theta_{\text{Task}}$

```
Input: target_data, train_data_loader, old_adapter, loss_fun, n, Task, $\Delta_{init}$, $\Delta_{lr}$
Output: Parameter changes $\Delta\Theta_{\text{Task}}$
if Task = IR then
    $b \leftarrow \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{IR}}}\mathcal{G}(x,y)$
else if Task = RC then
    $b \leftarrow \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{IM}}}\mathcal{G}(x,y) - \frac{1}{n}\sum_{(x+\delta_x,y)\in\mathcal{S}_{\text{IM}}}\mathcal{G}(x+\delta_x,y)$
else if Task = IM then
    $b \leftarrow \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_{\text{RC}}}\mathcal{G}(x,y) - \frac{1}{n}\sum_{(x,y+\delta_y)\in\mathcal{S}_{\text{RC}}}\mathcal{G}(x,y+\delta_y)$
end if
$\Delta \leftarrow$ initialize$(\Delta_{init})$
$optimizer \leftarrow$ Adam$([\Delta], lr = \Delta_{lr})$
while not converge do
    $data \leftarrow$ get_batch$(train\_data\_loader)$
    $batch\_loss \leftarrow$ loss_fun$(data.x, data.y)$
    $batch\_grad \leftarrow \nabla(batch\_loss, old\_adapter.parameters())$
    $hvp \leftarrow \nabla(batch\_grad, old\_adapter.parameters(),$ output $= b)$
    $optimizer.zero\_grad()$
    $funv\_value \leftarrow \frac{1}{2} \cdot \langle hvp, p\rangle - \langle b, p\rangle$
    $funv\_value.backward()$
    $optimizer.step()$
end while
Return Parameter changes $\Delta\Theta_{\text{Task}} = \Delta$
```

D EXPERIMENTAL DETAILS

D.1 THE TASKS FOR LLMS AND MLLMS

In this section, we briefly introduce the tasks used to validate LLMEraser on the unlearning tasks for IR, QM, and RC, as discussed in Section 4. These tasks are designed to assess LLMEraser's effectiveness in handling unlearning scenarios, where specific instances or data are removed or corrected when certain unlearning request arrives.

- For LLM4Rec unlearning tasks, our implementation is based on two representative PEFT methods: TallRec (Bao et al., 2023) for item rating, and LLaRA (Liao et al., 2024) for item ranking. Specifically, we frame the rating tasks (TallRec) as a binary classification problem, predicting whether or not the user prefers a target item. We employ AUC as the evaluation metric. For the ranking tasks (LLaRA), which recommend items to users from a candidate set, we utilize HitRatio@1 and ValidRatio to evaluate the relevance of recommended items among all candidates and the proportion of effective responses separately.
- In terms of MLLMs unlearning tasks, we focus on hard hallucination mining, *e.g.,* understanding of relation (Wu et al., 2024) and spurious biases (Ye et al., 2024). We structure the evaluation as binary or multi-choice classification problems, which aim to select the ground-truth from the noisy labels. Specifically, for relation understanding, we follow (Wu et al., 2024) to present the Recall, F1-Score, Precision, Classification accuracy, Yes ratio as the evaluation metrics. For spurious biases, we follow (Wu et al., 2024) to show the classification accuracy for 9 types of spurious correlations, which is Background (BG), Texture and Noise (TN), Co-occurring Objects (CO), Relative Size (RS), Colorization (Col.), Orientation (Ori.), Lighting and Shadows (LS), Perspective and Angle (PA), and Shape (Sha.).

Table 7: The hyperparameter settings and details for PEFT on each dataset or task.

Method / Dataset	LoRA r	LoRA alpha	LoRA dropout	Target modules	#Params	#Trainable params	Percentage of trainable params
TALLRec	8	16	0.05	’q proj’,’v proj’	6.7×10^{10}	4.2×10^{7}	0.06%
LLaRA	8	16	0.10	’k proj’, ’v proj’, ’q proj’, ’o proj’, ’gate proj’, ’up proj’, ’down proj’	6.7×10^{10}	1.9×10^{8}	0.30%
MM-SPUBENCH	8	16	0.05	’k proj’, ’v proj’, ’q proj’, ’o proj’, ’gate proj’, ’up proj’, ’down proj’	7.1×10^{10}	4.1×10^{8}	0.58%
R-BENCH	8	16	0.05	’k proj’, ’v proj’, ’q proj’, ’o proj’, ’gate proj’, ’up proj’, ’down proj’	7.1×10^{10}	4.1×10^{8}	0.58%

Datasets: Our experimental datasets for LLM4Rec unlearning tasks include three commonly used recommendation datasets: BookCrossing (Ziegler et al., 2005), MovieLens (Harper & Konstan, 2016), and LastFM (Cantador et al., 2011). We follow the data preprocessing and dataset partitioning as described in (Bao et al., 2023) and (Liao et al., 2024). For MLLMs unlearning tasks, we utilize MMSpuBench (Ye et al., 2024), and R-Bench (Wu et al., 2024) with the representative masked instances for evaluation, partitioning the data is into training (60%), validation (20%), and testing (20%) set.

Baselines: We carefully select the following methods for comparison. **Original**: The original model without unlearning modifications. **Retrain**: It retrains the adapters using the dataset after correction or removal. **SISA** (Sekhari et al., 2021): It divides the training data into disjoint shards and subsequently retrains sub-models (adapters) associated with the shards containing unlearning data. **RecEraser** (Chen et al., 2022): An enhancement of SISA, refining the aggregation strategy and taking into account collaborative signals during data partitioning. **Gradient Ascent**: It finetunes adapters using the reverse gradients of the deleted data. **E2URec** (Wang et al., 2024): An approach to implement instance removal based on KL divergence within a teacher-student framework.

D.2 IMPLEMENTATION DETAILS

All implementations are performed using Python 3.8 and PyTorch Lightning 1.8.6. All methods are run on a single Nvidia A100 GPU.

For R-Bench, the learning rate is set to $1e-4$, with a batch size of 16. The MLP model uses a learning rate of $2e-5$. During the unlearning phase, the parameters are specified as: $\Delta_{lr} = 1e-5$ and $\Delta_{init} = 1e-5$. In MMSPubench, the unlearning parameters are specified as: $\Delta_{lr} = 2e-5$ and $\Delta_{init} = 1e-5$.

For the TallRec model, we selected optimal training parameters based on the original work. During the unlearning phase, the parameters are configured as: $\Delta_{lr} = 5e-5$ and $\Delta_{init} = 5e-3$.

For the LLARA model, we selected optimal training parameters based on the original work. For the MovieLens dataset, the parameters are set as follows: Δ_{lr} is $1e-4$ and Δ_{init} is $1e-5$. For the LastFM dataset, the parameters are configured to Δ_{lr} at $1e-5$ and Δ_{init} at $1e-6$.

The hyperparameter settings and details for PEFT on each dataset or task can be found in Table 7.

E ESTIMATION ERRORS ANALYSIS OF LLMERASER

The approximation errors in LLMEraser consist of two primary components: first, the errors introduced by the Taylor expansion approximation in the derivation of the influence function, where high-order terms are neglected; and second, the errors arising from the new algorithm proposed in Section 3.3 in our manuscript for solving the inverse Hessian-vector product. We will conduct the error analysis in two parts accordingly.

E.1 ERRORS ANALYSIS FOR TAYLOR EXPANSION APPROXIMATION

Without loss of generality, we consider approximation error in Equation 6 of our submitted manuscript. In other words, we will analyze the error $\|\Delta\Theta(\epsilon) + \epsilon H_{\hat{\Theta}}^{-1}\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta})\|$.

The derivation below follows from (Zhang et al., 2022), where we assume that $H_{\hat{\Theta}}$ is invertiable. As we discussed in our manuscript, this can be guaranteed if the second-order sufficient condition holds at $\hat{\Theta}$.

Since $\hat{\Theta}_{\text{new}}(\epsilon)$ is an optimal solution to the perturbed loss function defined in Equation 5 in the submitted manuscript, we have

$$\nabla_{\Theta}R(\mathcal{Z};\hat{\Theta}_{\text{new}}(\epsilon)) + \epsilon\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta}_{\text{new}}(\epsilon)) = 0.$$

Since $\hat{\Theta}_{\text{new}}(\epsilon) \approx \hat{\Theta}$ when ϵ is sufficiently small, it follows from the Taylor expansion that

$$0 = [\nabla_{\Theta}R(\mathcal{Z};\hat{\Theta}) + \epsilon\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta})] + [H_{\hat{\Theta}} + \epsilon\nabla_{\Theta}^2\mathcal{L}((x,y);\hat{\Theta})]\Delta\Theta(\epsilon) + o(\|\Delta\Theta(\epsilon)\|).$$

Since $\hat{\Theta}$ is an optimal solution to the loss function defined in Equation 3 in the submitted manuscript, we have $\nabla_{\Theta}R(\mathcal{Z};\hat{\Theta}) = 0$. Therefore,

$$\Delta\Theta(\epsilon) = -[H_{\hat{\Theta}} + \epsilon\nabla_{\Theta}^2\mathcal{L}((x,y);\hat{\Theta})]^{-1}(\epsilon\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta}) + o(\|\Delta\Theta(\epsilon)\|).$$

Since $\hat{\Theta}$ is an optimal solution to the loss function defined in Equation 3 in the submitted manuscript, $H_{\hat{\Theta}}$ is positive semidefinite. Therefore, the assumption that $H_{\hat{\Theta}}$ is invertiable implies that $H_{\hat{\Theta}}$ is positive definte. Therefore, we know that

$$\Delta\Theta(\epsilon) = -H_{\hat{\Theta}}^{-1}(\epsilon\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta}) + o(|\epsilon|)\|\Delta\Theta(\epsilon)\| + o(\|\Delta\Theta(\epsilon)\|).$$

Therefore, as $\epsilon \to 0$,

$$\|\Delta\Theta(\epsilon) + H_{\hat{\Theta}}^{-1}(\epsilon\nabla_{\Theta}\mathcal{L}((x,y);\hat{\Theta})\| = o(|\epsilon|) + o(\|\Delta\Theta(\epsilon)\|) \to 0.$$

In our applications, we know that $\epsilon = O(1/n)$, where n is the number of training samples. Therefore, ϵ should be very small and our approximation to $\Delta\Theta(\epsilon)$ by the influence function should be accurate for applications with a very large training datasets.

E.2 ERRORS ANALYSIS FOR OUR PROPOSED ALGORITHM

For our proposed Algorithm, the estimation errors analysis is as follows. For a given (approximate) solution $\widetilde{\Delta}$ to the Equation 12 in our manuscript, the error is defined as

$$err(\widetilde{\Delta}) := \|\nabla_{\Theta}^2 R(\mathcal{Z};\widehat{\Theta})\widetilde{\Delta} - b\| = \|\nabla F(\widetilde{\Delta})\|,$$

where the function $F(\cdot)$ is defined in Equation 14 in the submitted manuscript. Therefore, the theoretical analysis of $err(\widetilde{\Delta})$ is equivalent to the error analysis of $\|\nabla F(\Delta_t)\|$ for the sequence $\{\Delta_t\}_{t\geq 1}$ generated by the optimization algorithm for solving the problem Equation 9, Equation 10, and Equation 11 in the submitted manuscript.

Since we use ADAM as a default optimizer for solving Equation 9, Equation 10, and Equation 11, we analyze the error $\|\nabla F(\Delta_t)\|$ for the sequence $\{\Delta_t\}_{t\geq 1}$ generated by ADAM. It follows from (Zhang et al., 2022) that ADAM can converge without modifications if the hyper-parameters are appropriately chosen (say the default choice $\beta_1 = 0.9$, $\beta_2 = 0.999$).

Table 8: Memory usage (measured in megabytes, MB) for different LoRA ranks (8, 16, 32) on the QM task, using LLaRA as the LLM4Rec model on the LastFM dataset, where 10% of users have items replaced with noisy interactions.

Method	LoRA r = 8	LoRA r = 16	LoRA r = 32
Retrain	33040 MB	33868 MB	34128 MB
SISA	33040 MB	33868 MB	34128 MB
LLMEraser (Ours)	**30760 MB**	**31386 MB**	**31834 MB**

Table 9: Experimental results on the QM task for different LoRA ranks (8, 16, 32), using LLaRA as the LLM4Rec model on the LastFM dataset, where 10% of users have items replaced with noisy interactions. ”Corrupted” refers to the model trained with the noisy data.

Method	LoRA r = 8		LoRA r = 16		LoRA r = 32	
	HitRatio@1	ValidRatio	HitRatio@1	ValidRatio	HitRatio@1	ValidRatio
Retrain	0.4508	1.0000	0.4417	0.9836	0.4215	0.9918
Corrupted	0.4344	0.9918	0.4098	1.0000	0.4016	1.0000
LLMEraser	**0.4426**	**1.0000**	**0.4344**	**1.0000**	**0.4180**	**1.0000**

Moreover, under reasonable assumptions (see (Zhang et al., 2022) for more details), it holds that

$$\min_{k_m \le t \le T} \mathbb{E}\|\nabla F(\Delta_t)\|_2 = \mathcal{O}(\log T/\sqrt{T}) = \widetilde{\mathcal{O}}(1/\sqrt{T}).$$

Since for sufficiently large T, $\log T < T^q$ for any $q > 0$, we know we can achieve

$$\min_{k_m \le t \le T} \mathbb{E}\|\nabla F(\Delta_t)\|_2 \le \epsilon$$

for small $\epsilon > 0$ in $\widetilde{\mathcal{O}}(\epsilon^{-2}) \approx O(\epsilon^{-2})$ iterations. This proof also ensures the convergence of the algorithm proposed in Section 3.3.

F DISCUSSION ABOUT THE EFFICIENCY OF LLMERASER

Our proposed algorithm in Section 3.3 for computing the parameter changes not only accelerates the calculation of parameter changes but also significantly reduces GPU memory consumption. As highlighted in our paper, while Conjugate Gradients (CG) is an effective method for computing parameter changes, it requires full-batch computation (Agarwal et al., 2016), which is infeasible for LLMs. Our new algorithm overcomes this limitation, making it practical to compute adapter's parameter changes in the context of LLMs.

Specifically, LLMEraser formulates the parameter updates as an inverse Hessian-vector product (Equation 9, Equation 10, and Equation 11 in our manuscript). Importantly, although the inverse Hessian appears in the formulation, it does not require explicit computation or inversion of the Hessian matrix. Directly calculating the inverse Hessian-vector product has a time complexity of $O(p^3)$ and a space complexity of $O(p^2)$, as the Hessian matrix needs to be stored—making it highly memory-intensive.

Our method transforms the computation of the inverse Hessian-vector product into the problem of solving for the Hessian-vector product, enabling efficient resolution through mini-batch algorithms. The Hessian-vector product, if computed directly via the full Hessian matrix multiplication, would have a time and space complexity of $O(p^2)$. However, using HVP (Hessian-free methods), we avoid the explicit computation and storage of the Hessian matrix, reducing both time and space complexity to $O(p)$ (Pearlmutter, 1994). By further leveraging mini-batch optimization for Equation 12 in the manuscript, LLMEraser achieves a space complexity of $O(p)$, ensuring its scalability.

The results for the LastFM dataset using the LLaRA backbone with LoRA ranks of 8, 16, and 32 are shown in the Table 9.

Table 10: Execution time (measured in seconds) for different LoRA ranks (8, 16, 32) on the QM task, using LLaRA as the LLM4Rec model on the LastFM dataset, where 10% of users have items replaced with noisy interactions.

Method	LoRA r = 8	LoRA r = 16	LoRA r = 32
Retrain	1.68×10^4	1.69×10^4	1.69×10^4
LLMEraser (Ours)	1.50×10^3	1.53×10^3	1.56×10^3

Table 11: Execution time (measured in seconds) of the IR task, using TALLRec as the LLM4Rec model on the BookCrossing dataset, where 5% of user interaction records are deleted.

	Retrain	Gradient Ascent	E2URec	**LLMEraser (Ours)**
Time (s)	5.6×10^3	2.3×10^3	2.4×10^2	4.9×10^1

We can observe that LLMEraser effectively reduces the negative impact of noisy data and brings a significant utility gain. The HitRatio@1 improves by an average of 4.9%, and the performance is comparable to that of Retrain. This demonstrates that LLMEraser can effectively forget and correct the adverse effects caused by noisy data.

Regarding GPU memory usage, we measure the GPU utilization of the LLaRA backbone with LoRA rank sets to 8, 16, and 32. The statistical information and the experimental results (with memory usage measured in megabytes (MB)) are shown in Table 8.

The GPU utilization of SISA is identical to that of Retrain because SISA (Kwak et al., 2017) effectively requires retraining all parameters (We report the memory usage required to train a single shard). Similarly, fine-tuning-based methods such as gradient descent also necessitates updating all parameters. The backbone of the LLM we used is LLaMA2-7B (Touvron et al., 2023b). Our method is highly memory-efficient and can be executed on a single A40 GPU.

The runtime results for LoRA with ranks 8, 16, and 32 on the LastFM dataset are shown in Table 10. The evaluation is measured in seconds (s). For TallRec, the execution times in the IR task is presented in Table 11.

In summary, the time and space complexity of LLMEraser are both $O(p)$, where p represents the number of parameters. This indicates that LLMEraser is highly efficient in terms of both time and space, as its performance scales linearly with the number of parameters. This efficiency makes LLMEraser a suitable choice for real-world applications where computational resources and time are critical considerations.

G PERFORMANCE OF LLMERASER ON THE GENERATIVE TASK

To further demonstrate the generalization ability of LLMEraser, we conduct additional experiments on other tasks, including dialogue summarization and question-answering tasks.

G.1 RESULTS OF THE INSTANCE REMOVAL (IR) TASK

For Instance Removal (IR) task, we conduct experiments on Alpaca-LoRA for the instruction tuning task. The backbone of LLM is LLaMA2-7B (Touvron et al., 2023b). We remove 5% of the training samples and evaluate the performance of the original model, retrained model, and LLMEraser on the deleted data using ROUGE scores (Chen et al., 2021) as the evaluation metric. The experimental results are shown in Table 12.

We can observe that LLMEraser closely matches the performance of Retrain. This is attributed to its direct estimation of parameter changes between the retrained model and the original model, enabling accurate calculations of these changes.

Table 12: Experimental results on the instance removal task with 5% of the training data removed, using Alpaca-LoRA for the instruction tuning task.

	ROUGE-1	ROUGE-2	ROUGE-L
Original	44.23	27.09	34.88
Retrain	39.84	20.40	31.78
LLMEraser	**39.55**	**20.17**	**31.63**

Table 13: Experimental results on the QM and RC tasks, using DIALOGSUM dataset. "50% Dialogue Distortion" refers to randomly removing dialogue tokens for 50% of the samples, while "20% Summary Distortion" refers to randomly removing summary tokens for 20% of the samples.

		ROUGE-1	ROUGE-2	ROUGE-L
50% Dialogue Distortion	Retrain	36.59	12.78	29.56
	Corrupted	34.09	11.61	26.66
	LLMEraser (Ours)	**35.92**	**12.22**	**28.98**
20% Summary Distortion	Retrain	36.59	12.78	29.56
	Corrupted	35.72	11.67	29.09
	LLMEraser (Ours)	**36.34**	**12.49**	**29.45**

G.2 RESULTS OF THE QUERY MODIFICATION (QM) AND RESPONSE CORRECTION (RC) TASKS

For Query Modification (QM) and Response Correction (RC) tasks, we supplement experimental results on the DIALOGSUM (Chen et al., 2021) dataset for dialogue summarization. The backbone of the LLM is FLAN-T5 Base (Raffel et al., 2020). Specifically, we apply perturbations to 50% of the samples in the dialogue and 20% of the samples in the summary, and use LLMEraser to correct the corrupted data. The experimental results are presented in Table 13.

For MLLM, we conduct experiment on the mm-vet-v2 dataset for integrated capability evaluation task (Yu et al., 2024). The data format of mm-vet-v2 is actually in the form of text-based question-answer pairs. The backbone of MLLM is LLaVA 1.5-7B. Specifically, we randomly select 80% of mm-vet-v2 samples as training set, and employ the left 20% samples for evaluation. We apply perturbations to 50% of the training samples and evaluated the performance of the retrained model, corrupted model and LLMEraser on the testing set, where LLMEraser corrects the corrupted data. Here are the experimental results on the Query Modification (QM) task, where we utilize "rec, gen, ocr, spat, know" capacities for evaluation, and report the average results. All Experimental scores are calculated with gpt-4-turbo by following (Yu et al., 2024). The results are presented in Table 14.

LLMEraser provides a substantial utility improvement to the model compared to the corrupted baseline, effectively mitigating the negative impact of noisy data. On the LLM dialogue summarization task, LLMEraser achieves an average improvement of 6.45% on QM tasks and 3.34% on RC tasks compared to the corrupted baseline. For MLLM QA tasks, LLMEraser achieves an average improvement of 12.35% on QM task. Furthermore, LLMEraser's performance is close to that of Retrain, highlighting its effectiveness in correcting inaccurate information.

H RELATED WORK

H.1 LARGE LANGUAGE MODELS

Recent advancements in natural language processing (NLP) (Nam et al., 2024; Jin et al., 2024) have been significantly driven by the development of pretrained language models and Large Language Models. The introduction of models like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) marked a pivotal shift in leveraging large-scale unsupervised pretraining, enabling superior performance across various NLP tasks through fine-tuning. The scaling of language models led to the emergence of LLMs such as GPT-3 (Brown et al., 2020) and PaLM (Chowdhery et al., 2023),

Table 14: Experimental results on the QM tasks, using LLaVA 1.5-7B as the backbone model on the mm-vet-v2 dataset.

	rec	gen	ocr	spat	know	Average
Retrain	37.8	28.9	28.3	37.7	21.6	30.9
Corrupted	29.4	23.0	20.7	34.1	14.1	24.3
LLMEraser (Ours)	**33.1**	**27.0**	**21.5**	**37.2**	**17.5**	**27.3**

which have pushed the boundaries of language understanding and generation. These models, with billions of parameters, are capable of performing complex reasoning and handling diverse tasks with minimal instruction.

Recent research has explored parameter-efficient fine-tuning techniques, which adapt large models to specific applications without requiring extensive computational resources. Techniques like Adapter modules (Houlsby et al., 2019) and Low-Rank Adaptation (LoRA) (Hu et al., 2022) have gained popularity for their efficiency and effectiveness in maintaining performance while reducing the number of trainable parameters. Furthermore, instruction tuning (Liu et al., 2023a; Tang et al., 2024) using domain-specific data has emerged as a key strategy to enhance model performance in specialized contexts. Works by Ouyang et al. (2022) and Dodge et al. (2020) illustrate how tailoring models to specific tasks through targeted instruction can significantly improve their utility, particularly in complex domains, demonstrating the importance of context and relevance in model training.

LLMs have found extensive applications in various downstream tasks, demonstrating their versatility across domains such as natural language processing, information retrieval, and knowledge graph augmentation (Hu et al., 2024a; Zhang et al., 2024a; Xu et al., 2024b). For instance, LLMs are employed to enhance the accuracy of query-based systems by leveraging their ability to understand and generate contextually relevant responses, improving user experience in search applications (Liu et al., 2024b; Shang & Huang, 2024). Additionally, they are utilized in graph analytics, enabling complex reasoning tasks and facilitating the extraction of insights from structured data (Chen et al., 2023; Xu et al., 2024a). The adaptability of LLMs through prompt engineering further supports their deployment in specific use cases, allowing for tailored outputs that meet diverse requirements (Arawjo et al., 2024; Cain, 2024).

H.2 LARGE LANGUAGE MODELS UNLEARNING

The concept of unlearning in Large Language Models has garnered considerable attention as concerns over data privacy and model integrity have intensified. In-context unlearning, proposed by Pawelczyk et al. (2023), allows the selective removal of data points by supplying flipped labels during inference, effectively maintaining performance while unlearning specific information. Additionally, Quark by Lu et al. (2022) employs a reinforcement learning framework to control and reduce undesirable behaviors, enhancing text generation without extensive retraining.

Chen & Yang (2023) introduce a lightweight unlearning method that integrates unlearning layers into transformer architectures, facilitating efficient data removal. Knowledge Unlearning by Jang et al. (2023) demonstrates that targeted gradient ascent can effectively forget sensitive information, surpassing traditional methods in performance retention. The technique proposed by Eldan & Russinovich (2023) facilitates the removal of specific facts related to the Harry Potter series while preserving the model's overall performance.

Other approaches, such as the Partitioned Gradient Update (PGU) method by Yu et al. (2023), aim to reduce social biases effectively. Collectively, these studies underline the significance of unlearning in LLMs, paving the way for safer, more responsible AI applications.