UnReL: Efficient Exact LLM Unlearning via Targeted Relearning

Tamim Al Mahmud[0000−0002−3633−4921], David Sánchez[0000−0001−7275−7887], and Josep Domingo-Ferrer[0000−0001−7213−4962]

Universitat Rovira i Virgili
Department of Computer Engineering and Mathematics
ComSCIAM-Center for Computational Science and Applied Mathematics
CYBERCAT-Center for Cybersecurity Research of Catalonia
Av. Països Catalans 26, 43007 Tarragona, Catalonia
{tamimal.mahmud, david.sanchez, josep.domingo}@urv.cat

Abstract. The growing integration of large language models (LLMs) in various applications raises concerns about data privacy and the need for efficient unlearning. Exact unlearning methods are computationally expensive and can suffer from utility degradation, while approximate approaches do not guarantee sufficient data removal to comply with data privacy regulations. We introduce UnReL (Unlearning via Targeted Relearning), a framework for efficient and exact unlearning in LLMs. UnReL combines heuristic data sharding, based on the likelihood of future forgetting requests, with targeted relearning to minimize retraining costs while ensuring regulatory compliance. Experimental results demonstrate that UnReL outperforms existing unlearning methods in both computational efficiency and model utility retention. Our code is available at https://github.com/tamimalmahmud/LLM-Unlearning/tree/main/UnReL

Keywords: Data privacy · Right to be forgotten · Machine unlearning · Large language models

1 Introduction

Large language models (LLMs) are trained on enormous datasets that often include both public and private data, with web-scale data typically serving as the primary source. As a result, LLMs can store personal data [1] and copyrighted material [2], which might be disclosed when users interact with models [1]. To comply with regulations such as the GDPR [3] in the EU and the CCPA [4] in the US, model administrators must be able to remove this sensitive information from trained models on request to enforce the *right to be forgotten* (RTBF).

The straightforward approach to digital forgetting is to retrain the model from scratch, excluding unwanted information. However, this approach is computationally unfeasible for LLMs. To address this, machine unlearning [5–7] has been proposed as a more efficient alternative. *Exact unlearning* [8–10] modifies model training to facilitate unlearning while ensuring the removal of unwanted

data, but it still incurs high computational costs and compromises model utility. In contrast, *approximate unlearning* [11, 7, 12] modifies trained models *ex post*, *e.g.*, by adjusting model parameters. Although this approach is more efficient, it does not *guarantee* complete removal of unwanted data and thus falls short of satisfying the RTBF.

To address these challenges, we present UnReL, a framework for efficient exact unlearning. Rather than introducing a new unlearning objective, UnReL defines an unlearning-ready training protocol based on heuristic sharding, cumulative checkpoints, and targeted relearning. It assigns training data samples to the shards according to a heuristic estimate of their likelihood of being affected by future forgetting requests. After incremental training, UnReL saves a checkpoint after each shard. When a forgetting request arrives, it rolls back to the last unaffected checkpoint and retrains only the affected and subsequent shards. Since higher-risk data samples are placed in later shards, the expected retraining cost is reduced.

Our experiments show that UnReL can unlearn data much faster than other exact unlearning methods while better retaining the utility of the model.

The remainder of the paper is organized as follows. Section 2 discusses related work on machine unlearning. Section 3 presents our UnReL framework. Section 4 describes the experimental setup. Section 5 reports and discusses the results. The last section provides conclusions and several lines of future research.

2 Related work

Training a new model from scratch without the unwanted data is a straightforward approach to digital forgetting, but it is impractical for any large model, let alone LLMs. *Machine unlearning* has been proposed as a more practical alternative. We can distinguish two primary types of unlearning methods: *exact unlearning* and *approximate unlearning*.

Exact unlearning methods guarantee complete data removal from models, but incur significant computational costs and potential utility degradation. A foundational approach, SISA (Sharded, Isolated, Sliced, and Aggregated) [8], reduces retraining costs by partitioning training data into shards, training separate models for each shard, and performing incremental training on each slice. This approach ensures that only the shards that contain unwanted data are retrained. However, the main drawback of SISA is that as the number of shards increases, model accuracy can degrade significantly, as training partial models on isolated shards limits the model's ability to generalize. Similarly, APA (Adapter Partition and Aggregation) [9] reduces unlearning costs by partitioning data into disjoint shards and retraining only the affected adapters in LLM-based recommendation systems. However, when aggregating parameters from multiple adapters, misalignment of outputs can lead to the loss of critical information, further compromising model performance.

Approximate unlearning techniques are more affordable because they adjust model parameters *ex post*, rather than retraining. For example, Gradient Ascent

(GA) [16, 7] increases loss at unwanted data points, making the model less likely to retain or repeat that information. Gradient Difference (GD) [11] merges gradient ascent and descent in an effort to find the right balance between forgetting unwanted data and keeping unrelated information. Kullback-Leibler (KL) divergence [12] preserves crucial knowledge by narrowing the gap between a model's prior distributions before and after unlearning. Although these techniques are much more efficient than exact unlearning methods above, they do not guarantee forgetting and may remove useful not-to-be-forgotten knowledge.

Recent LLM unlearning studies have also explored alternative mechanisms, such as data extraction after exact unlearning [13], modular retain-free unlearning [14], and neural activation redirection [15]. While these studies contribute to broader threat models, they still face challenges related to efficiency or the inability to guarantee complete data removal.

3 The UnReL framework

UnReL presents a novel two-phase framework for efficient and exact unlearning in LLMs. Unlike traditional exact unlearning methods, which rely on random data sharding or approximate methods that adjust model parameters post-training, UnReL utilizes a risk-aware heuristic to partition data and performs incremental training with cumulative checkpoints. This approach not only minimizes retraining costs, but also ensures compliance with privacy regulations by guaranteeing the removal of unwanted data. The framework considers two connected phases, (A) *unlearning-ready training* and (B) *unlearning via targeted relearning*, as shown in Figure 1.

Phase (A): Unlearning-ready training. We partition training data ($\mathcal{D}$) into public $\mathcal{P}$ (*e.g.* Wikipedia, open web content, permissively licensed content, etc.) and non-public $\mathcal{R}$ (*i.e.* restricted, private, or copyrighted data). By definition, public data do not require forgetting, so we place them in a single shard ($S_0 := \mathcal{P}$).

To reduce the overhead of future unlearning, UnReL splits the non-public dataset $\mathcal{R}$ into smaller shards based on their likelihood of being affected by future forgetting requests. This likelihood is operationalized as a heuristic forgetting score $L(x)$, which is used to rank data samples before shard assignment and combines two factors: content risk $f_r(x)$ and freshness $f_f(x)$.

1. *Risk:* Let $f_r(x)$ represent the probability that a data sample x includes content that could motivate a forgetting request, such as personal information, health or financial data, or copyright-restricted material. Rather than interpreting this value as a calibrated probability, we estimate $f_r(x)$ as a value in $[0, 1]$ using the semantic similarity and explicit detection signals. Each data sample x is converted into an embedding vector $e(x) = \phi(x)$ using a pre-trained sentence-transformer model, which captures its meaning in a continuous space. To estimate how sensitive x is, we measure the cosine similarity between $e(x)$ and a fixed set of sensitive prototype embeddings $\{\mu_c\}$ representing medical, personal-identity, and financial domains. Across all tests,

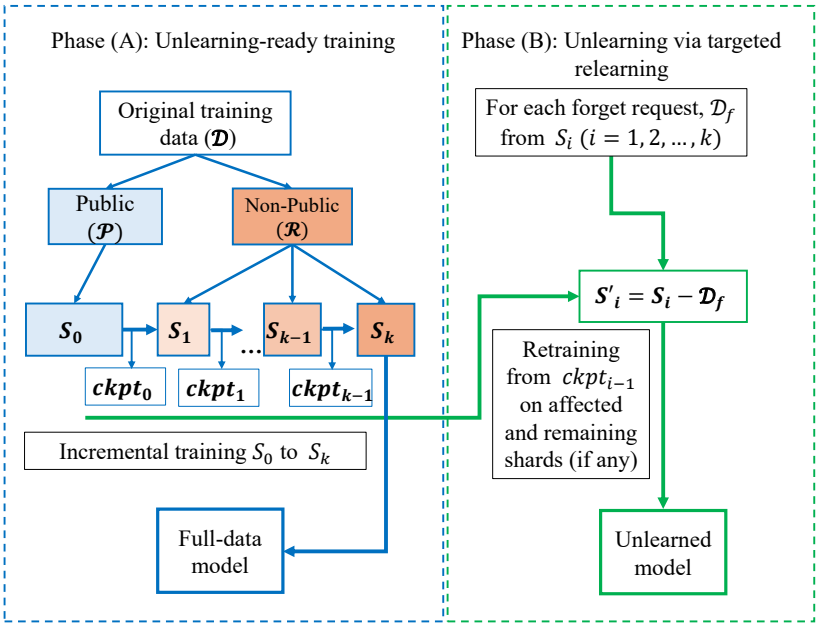


Fig. 1: The UnReL Framework

these prototypes stay constant and are constructed as mean embeddings of carefully curated domain descriptor phrases. The semantic component is computed as the maximum normalized cosine similarity across the prototype set:

$$f_{\text{sem}}(x) = \max_{c} \frac{1 + \cos\big(e(x), \mu_c\big)}{2}. \tag{1}$$

In addition to semantic similarity, we detect explicit privacy indicators using regular expressions, such as email addresses, phone numbers, IP addresses, IBAN-like patterns, debit/credit card numbers validated with the Luhn checksum, and generic identifiers near identity-related keywords. The final content-risk score $f_r(x)$ is calculated as a convex combination of semantic and explicit components:

$$f_r(x) = \gamma\, f_{\text{sem}}(x) + (1 - \gamma)\, f_{\text{exp}}(x). \tag{2}$$

This hybrid design minimizes the reliance on a single detection technique while increasing tolerance to fluctuation in the vocabulary.

2. *Freshness:* In addition to content sensitivity, we also model exposure risk induced by recency. Forgetting requests for recent data samples are more frequent than those for older data samples, according to operational data governance protocols [17]. We estimate this probability using a temporal

decay function applied to the publication age of the data sample, as derived from available metadata (*i.e.* publication date, timestamp, or inferred year):

$$f_f(x) = e^{-\lambda \, \mathrm{age}(x)}, \quad \lambda > 0. \tag{3}$$

Here, $f_f(x)$ denotes the probability of a forgetting request due to recency, and λ determines the rate at which this probability decreases with the age of the data.

The general likelihood of forgetting for a data sample x is defined as:

$$L(x) = w_r \, f_r(x) + w_f \, f_f(x), \quad w_r, w_f \geq 0, \; w_r + w_f = 1. \tag{4}$$

UnReL uses $L(x)$ to rank data samples for shard assignment and reduce the expected retraining cost. $L(x)$ is not a calibrated probability but a heuristic ranking score used to optimize shard ordering. If a request affects an earlier shard, UnReL still removes the target data and continues learning from the last unaffected checkpoint.

Before shard assignment, all risk and freshness scores are standardized from min to max throughout the dataset. A sensitivity study was performed for various weight configurations for consistent shard ordering, demonstrating that UnReL does not rely on a specific weighting option.

Once the likelihood $L(x)$ of a forgetting request is calculated for each training data, UnReL sorts $\mathcal{R}$ by ascending values of $L(x)$ and partitions it into k disjoint shards $S_1, S_2, \ldots, S_k$. The sizes of the shards are not necessarily the same; *e.g.*, one might decide to include in a shard all data that have a similar value of L. With this approach, the earlier shards contain older, lower-risk data, while the later shards store newer, more sensitive data.

The model is then trained incrementally from S_0 to S_k, saving a cumulative checkpoint after each shard (*i.e.*, $ckpt_i$ from S_i). Each checkpoint stores all the knowledge learned up to that point, making it immediate to roll back if forgetting of some data stored after a specific checkpoint is needed. The last checkpoint corresponds to the full data model. For incremental learning, we used the continual learning approach from Red Hat AI and IBM Research [18], which uses adaptive singular value decomposition and orthogonal subspace learning to prevent catastrophic forgetting.

Phase (B): Unlearning via targeted relearning. When a forgetting request for data $\mathcal{D}_f$ arrives, UnReL identifies the shard (*i.e.*, S_i) containing those data. The requested data within shard S_i are removed, producing an updated shard S_i'. Afterward, the model is retrained solely on the updated shard S_i' and $S_{i+1}, \cdots, S_k$, starting from checkpoint $ckpt_{i-1}$. This produces an updated unlearned model that guarantees forgetting of unwanted data by design, similarly to retraining from scratch on the retained data but at substantially lower computational cost.

The framework naturally supports batch processing of multiple forgetting requests. When multiple forgetting requests are received within a specified timeframe (*e.g.*, processing on a monthly basis), all impacted data samples can be

removed from their corresponding shards before starting retraining. In this scenario, retraining is initiated from the earliest affected shard, thus spreading the unlearning expense across multiple requests.

4 Experimental setup and evaluation protocol

Compared methods. For comparison, we reproduced the exact methods (SISA and APA) and approximate techniques (GA, GD, and KL) described in Section 2. Moreover, we used *retraining from scratch on the retain set* (RFS-R) as a reference baseline for unlearning.

Dataset and models. To simulate the scenario we tackle, we used two pre-trained text generation models of different sizes –Phi-1.5B and Qwen3-4B– because training an LLM from scratch is hardly affordable. To control the data used for training (and thus susceptible to requiring unlearning), we fine-tuned these models on the TOFU dataset [7], a recent benchmark for evaluating LLM unlearning. TOFU contains 4,000 question–answer pairs generated from 200 synthetic author profiles. The synthetic nature of the data ensures that there is no overlap with the pre-training corpora of the models, and thus enables an unbiased evaluation. It also includes two real-world subsets, *Real Authors* and *Real-World Facts*, for evaluating model performance and generalization pre/post unlearning.

As described in Section 3, both the retain and forget subsets in $\mathcal{R}$ were sorted according to their estimated likelihood $L(x)$ of requiring forgetting, calculated as a weighted combination of risk and freshness. To validate the effectiveness of the system, we tested it on datasets containing a mix of general and sensitive content, including personal, medical, financial and other sensitive data. This validation demonstrated that the system effectively identifies privacy-sensitive content, assigning higher $L(x)$ scores for data more likely to contain sensitive information. During this validation, we tested several configurations for the risk (w_r) and freshness (w_f) weights and ultimately set $w_r = 0.7$ and $w_f = 0.3$, prioritizing content sensitivity over recency. This configuration provided the best balance, confirming consistent and robust performance, and ensuring effective prioritization of content for unlearning. More generally, these parameters can be adapted to the requirements of a particular application (including applicable legal frameworks of the RTBF). Thus, they add flexibility and adaptability to the system. The choice of their values in our experiments is just a reasonable example, but by no means a recommendation based on representative empirical evidence.

The TOFU forget set comprised 20 author profiles. For UnReL, the likelihood that each profile would require forgetting was calculated, and the profiles were distributed across 2 and 4 equal-sized shards in $\mathcal{R}$ according to their scores. In contrast, for SISA and APA, profiles were assigned to shards using the random seeding that these methods employ. The corresponding log is available.[1] For evaluation, we selected the five authors from the forget set (corresponding to

[1] https://github.com/tamimalmahmud/LLM-Unlearning/tree/main/UnReL/CalculateL_x/forgetdata

2.5% of the total training data) most likely to require forgetting. As expected, UnReL placed them deterministically in S_2 for 2 shards and S_4 for 4 shards, but SISA and APA placed them randomly across shards.

Similarly, data samples from the retain set were allocated to the shards according to their likelihood of being forgotten. Each shard contained an equal number of author profiles. The TOFU custom-split datasets used in this study are available.[2]

Evaluation metrics. Following the TOFU evaluation strategy [7], we considered the following two metrics:

- Model utility (MU) measures how well the model preserves useful knowledge after unlearning. MU is evaluated using ROUGE-L (RL), Conditional Probability (CP), and Truth Ratio (TR). RL checks how much the model's output matches the ground truth, CP measures the confidence of the model in its answer, and TR compares the chances of giving incorrect versus correct paraphrased answers. Each metric is computed on the retain set, the *Real Authors* and the *Real-World Facts* datasets, then normalized to $[0, 1]$, and aggregated via the harmonic mean to produce an overall MU score.
- Forget quality (FQ) is measured by comparing truth-ratio distributions between unlearned and retain-only (RFS-R) models using the Kolmogorov Smirnov (KS) test. A KS test p-value below 0.05 indicates poor forgetting: the unlearned model does not match the retain-only model. In contrast, the greater p the closer the match is, which means effective forgetting. The threshold 0.05 is a widely accepted standard for statistical significance [7]. FQ metrics complementary to KS are also reported in the next section.

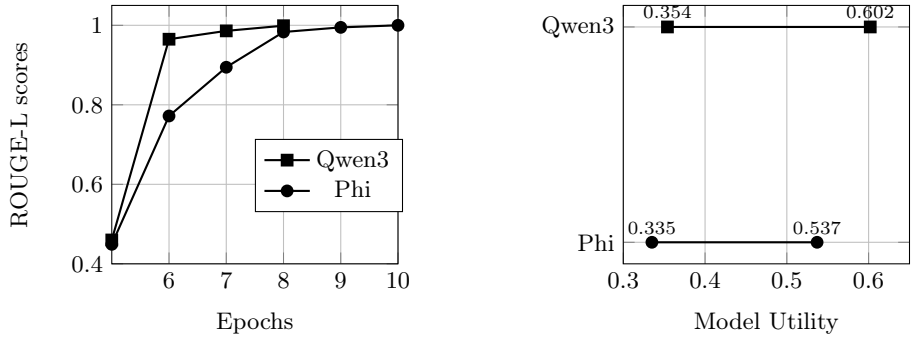
Training settings. We empirically observed that the **Phi-1.5B** model, when trained on the full TOFU dataset for $E = 10$ epochs, achieved a perfect ROUGE-L ($RL = 1.0$) and a maximum model utility ($MU = 0.537$), representing a substantial improvement over the pre-trained Phi baseline ($RL = 0.449$, $MU = 0.335$). Similarly, the **Qwen3-4B** model, trained for $E = 8$ epochs on the full TOFU dataset, attained a perfect ROUGE-L ($RL = 1.0$) and a peak model utility ($MU = 0.603$), compared to its pre-trained baseline ($RL = 0.460$, $MU = 0.354$). These results are summarized in Figure 2.

These observations guided our choice to train the Phi-1.5B model for $E = 10$ and the Qwen3-4B model for $E = 8$ epochs in both phases of UnReL. We used identical configurations for all baseline methods to ensure a fair comparison. All experiments were conducted on an NVIDIA H100 GPU equipped with 80 GB HBM3 memory, using a learning rate of 5×10^{-5}, a weight decay of 0.01, and an effective batch size of 16.

5 Results and discussion

We evaluated the exact unlearning methods –SISA, APA, and UnReL– with $|S| = 2$ and $|S| = 4$ shards in the private segment $\mathcal{R}$. For reference, the model

[2] https://huggingface.co/datasets/talmahmud/tofu_custom_split_UnReL



(a) ROUGE-L across epochs (b) MU (pre-trained → fine-tuned)

Fig. 2: ROUGE-L and model utility for Phi1.5B and Qwen3-4B models before and after training on the full TOFU dataset.

utility of Phi and Qwen3 after fine-tuning on TOFU full data is 0.537 and 0.603, respectively, as shown in Figure 2(b), which is the upper bound model utility before unlearning.

Model utility and forget quality. Table 1 reports MU and FQ (p-value) across models, different forgetting methods, shard configurations, and an increasing amount of data to be forgotten (*i.e.*, from 0.5% to 2.5%).

In particular, all approximate methods failed to meet the minimum forgetting threshold (0.05) and exhibited a sharp decline in MU, as they tend to over-forget, thus removing too much (useful) information. In contrast, exact unlearning methods achieve higher FQ and MU. UnReL achieved the best results, while SISA also tackled severe utility degradation through incremental slice training rather than individual training. APA experienced significant utility degradation as a result of individual shard training. Although SISA achieved moderate FQ and MU scores, its limitation in model generalization is evident from the FQ value before forgetting in the full-data model (marked in red). APA, which employs individual training, exhibits this limitation even more. On the other hand, UnReL's incremental learning approach, utilizing adaptive singular value decomposition and orthogonal subspace learning to build a single model, prevents forgetting of prior shards, leading to superior MU.

To complement the KS-based FQ score, we also report Jensen–Shannon Divergence (JSD) and absolute entropy difference (ΔH) between the truth-ratio distributions of UnReL and RFS-R in Table 2. Lower values indicate stronger resemblance to RFS-R and thus more effective forgetting. As shown in Table 2, both JSD and ΔH remain small, indicating that UnReL remains distributionally close to RFS-R beyond the KS-based FQ score.

Computational cost. Table 3 reports the training runtimes for all exact methods. Not surprisingly, they are slower than RFS-R, which only fine-tunes on new data without the overhead of preserving prior knowledge. UnReL incurs a higher training cost due to the use of adaptive singular value decomposition and orthogonal subspace learning to retain previous shard

Table 1: MU and FQ (p-value) for different unlearning methods. Before forgetting, a lower FQ is best as it indicates better learning. After forgetting, a higher FQ is best. The average best values are **bolded**.

Model	Methods	Full-data model (200 authors)				Forgotten authors												Average					
						0.5%				1%				2.5%									
		MU ↑		FQ ↓		MU ↑		FQ ↑		MU ↑		FQ ↑		MU ↑		FQ ↑		MU ↑		FQ ↑	
		|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4	|S|=2	|S|=4
Phi-1.5B	UnReL	0.506	0.502	0.002	0.005	0.491	0.488	0.832	0.913	0.488	0.478	0.679	0.762	0.489	0.487	0.682	0.908	**0.490**	**0.484**	**0.731**	**0.861**
	SISA	0.493	0.487	0.182	0.192	0.488	0.478	0.832	0.571	0.480	0.474	0.366	0.266	0.486	0.472	0.715	0.768	0.485	0.474	0.638	0.535
	APA	0.401	0.392	0.201	0.247	0.402	0.393	0.336	0.571	0.402	0.393	0.165	0.165	0.402	0.394	0.024	0.006	0.402	0.393	0.175	0.247
	GA	NA		NA		0.272		0.011		0.000		0.003		0.000		0.000		0.091		0.005	
	GD	NA		NA		0.446		0.105		0.440		0.029		0.269		0.000		0.385		0.045	
	KL	NA		NA		0.275		0.011		0.000		0.014		0.000		0.000		0.092		0.008	
	RFS-R	0.537		0.000		0.534		1.000		0.533		1.000		0.530		1.000		0.532		1.000	
Qwen3-4B	UnReL	0.560	0.535	0.012	0.002	0.569	0.534	0.776	0.929	0.572	0.546	0.535	0.638	0.561	0.533	0.377	0.803	**0.567**	**0.537**	**0.562**	**0.790**
	SISA	0.557	0.531	0.030	0.053	0.552	0.533	0.756	0.569	0.558	0.534	0.231	0.152	0.558	0.540	0.191	0.165	0.556	0.535	0.392	0.295
	APA	0.476	0.465	0.569	0.415	0.485	0.465	0.415	0.415	0.476	0.473	0.152	0.019	0.476	0.463	0.055	0.003	0.479	0.467	0.2073	0.1457
	GA	NA		NA		0.015		0.003		0.000		0.000		0.000		0.000		0.005		0.001	
	GD	NA		NA		0.462		0.011		0.494		0.000		0.486		0.000		0.480		0.004	
	KL	NA		NA		0.014		0.010		0.000		0.000		0.000		0.000		0.005		0.003	
	RFS-R	0.603		0.000		0.602		1.000		0.595		1.000		0.593		1.000		0.596		1.000	

Table 2: Complementary distributional tests for UnReL at $|S| = 2$.

Model	0.5%		1%		2.5%	
	JSD	ΔH	JSD	ΔH	JSD	ΔH
Phi-1.5B	0.0087	0.0588	0.0091	0.1189	0.0104	0.1064
Qwen3-4B	0.0095	0.0984	0.0163	0.1206	0.0157	0.1398

Table 3: Training runtime for exact unlearning methods

Model	Method	|S|=2	|S|=4
Phi-1.5B	UnReL	$46m58s$	$47m57s$
	SISA	$39m22s$	$41m33s$
	APA	$40m22s$	$40m45s$
	RFS-R	$31m30s$	
Qwen3-4B	UnReL	$51m19s$	$52m05s$
	SISA	$46m00s$	$47m04s$
	APA	$48m29s$	$48m45s$
	RFS-R	$38m56s$	

knowledge during incremental training. In contrast, SISA performs incremental training only on the slices (*i.e.*, shards in $\mathcal{R}$), whereas the public shard's traditional training reduces cost. While APA has a lower computing cost than UnReL, it is still higher than traditional end-to-end training, as each shard is optimized separately, causing convergence delays.

Table 4: Average unlearning time over five forget operations. "UnReL" reports the score-aligned setting, while "UnReL (non-ideal)" reports forced earlier-shard requests. SISA and APA results are averaged across shard configurations to ensure fair comparison rather than relying on a single random assignment.

Model	Method	|S|=2	|S|=4
Phi-1.5B	UnReL	$7m25s$	$3m55s$
	UnReL (non-ideal)	$11m15s$	$5m53s$
	SISA	$12m45s$	$11m40s$
	APA	$8m20s$	$4m12s$
	GA	$2m12s$	
	GD	$2m31s$	
	KL	$2m21s$	
	RFS-R	$31m30s$	
Qwen3-4B	UnReL	$9m49s$	$5m9s$
	UnReL (non-ideal)	$13m55s$	$7m22s$
	SISA	$15m34s$	$13m50s$
	APA	$10m50s$	$5m30s$
	GA	$7m11s$	
	GD	$7m44s$	
	KL	$7m08s$	
	RFS-R	$36m49s$	

Table 4 reports the runtimes of the unlearning phase for all the evaluated methods. For $|S| = 4$, APA and UnReL runtimes are comparable to those of the approximate methods, even better for the larger Qwen3 model. However, comparing exact and approximate methods is unfair, as the latter do not provide forgetting guarantees. Although APA demonstrates an unlearning runtime similar to UnReL, this benefit is offset by a substantial reduction in model performance.

In other words, efficiency is irrelevant if unlearning does not yield good utility and forget quality. This is also applicable to approximate unlearning methods.

Table 4 also reports a non-ideal scenario where forgetting affects earlier shards, such as S_1 for $|S| = 2$ and S_3 for $|S| = 4$. In this case, UnReL's runtime naturally increases as more shards need retraining. In the worst-case scenario, where forgetting requests do not align with $L(x)$ and affect earlier shards, UnReL's unlearning cost may approach that of SISA. However, UnReL still outperforms SISA by preserving higher model utility, as demonstrated in Table 1. This is due to UnReL's use of incremental training with cumulative checkpoints, which preserves a single model and avoids the overfitting risk of training separate models for each shard, as in SISA. The heuristic score improves unlearning efficiency, while the incremental design ensures better utility retention, even under less favorable conditions.

6 Conclusions and future work

We proposed UnReL, a framework for regulation-compliant exact unlearning in LLMs that leverages risk-aware heuristic data sharding and targeted relearning. Experiments on the TOFU benchmark show that UnReL achieves the highest forget quality and model utility while remaining computationally efficient.

As limitations of this work, we acknowledge that the empirical evaluation relies on the TOFU benchmark dataset that, while enabling controlled experimentation, may not comprehensively capture the heterogeneity and dynamic nature of real-world forgetting requests. Furthermore, UnReL requires checkpoint storage comparable to SISA and APA, leading to additional memory overhead. Despite these limitations, the proposed framework demonstrates that efficient, exact, and scalable unlearning in LLMs is practically attainable.

Future research will extend UnReL to larger models and diverse datasets, integrate precise risk estimation, and explore adaptive retraining strategies to balance regulatory compliance with unlearning at scale.

Acknowledgments. Partial support has been received from the Government of Catalonia (ICREA Acadèmia Prizes to J. Domingo-Ferrer and to D. Sánchez), MCIN/AEI under grant PID2024-157271NB-I00 "CLEARING-IT", and the European Commission (project HORIZON-101292277 "SoBigDataIP").

References

1. Lukas, N., Salem, A., Sim, R., Tople, S., Wutschitz, L., Zanella-Beguelin, S.: Analyzing Leakage of Personally Identifiable Information in Language Models. In: 2023 IEEE Symposium on Security and Privacy (SP), pp. 346–363. IEEE Computer Society, Los Alamitos, CA, USA (2023)
2. Karamolegkou, A., Li, J., Zhou, L., Søgaard, A.: Copyright Violations and Large Language Models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 7403–7412. ACL, Singapore (2023)

3. Voigt, P., Von dem Bussche, A.: The EU General Data Protection Regulation (GDPR). In: A Practical Guide, 1st Ed., pp. 10–5555. Springer International Publishing, Cham (2017)
4. Department of Justice, CCPA: California Consumer Privacy Act (CCPA) State of California - Department of Justice - Office of the Attorney General. Available at: https://oag.ca.gov/privacy/ccpa, last accessed 01/02/2026
5. Jang, J., Yoon, D., Yang, S., Cha, S., Lee, M., Logeswaran, L., Seo, M.: Knowledge Unlearning for Mitigating Privacy Risks in Language Models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, pp. 14389–14408. ACL, Canada (2023)
6. Yao, Y., Xu, X., Liu, Y.: Large Language Model Unlearning. In: Advances in Neural Information Processing Systems, Vol. 37, pp. 105425–105475. Curran Associates, Inc., USA (2024)
7. Maini, P., Feng, Z., Schwarzschild, A., Lipton, Z.C., Kolter, J. Z.: TOFU: A Task of Fictitious Unlearning for LLMs. In: Secure and Trustworthy LLM Workshop, ICLR 2024, pp. 1–25, USA (2024)
8. Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., Papernot, N.: Machine Unlearning. In: IEEE Symposium on Security and Privacy, pp. 141–159. IEEE, (2021)
9. Hu, Z., Zhang, Y., Xiao, M., Wang, W., Feng, F., He, X.: Exact and Efficient Unlearning for Large Language Model-Based Recommendation. In: IEEE Transactions on Knowledge and Data Engineering, Vol. 37, No. 10, pp. 5866–5877 (2025)
10. Al Mahmud, T., Jebreel, N., Domingo-Ferrer, J., Sánchez, D.: DP2Unlearning: An Efficient and Guaranteed Unlearning Framework for LLMs. In: Neural Networks, Vol. 192, pp. 107879 (2025)
11. Liu, B., Liu, Q., Stone, P.: Continual Learning and Private Unlearning. In: Conference on Lifelong Learning Agents, pp. 243–254. PMLR, USA (2022)
12. Yao, J., Chien, E., Du, M., Niu, X., Wang, T., Cheng, Z., Yue, X.: Machine Unlearning of Pre-trained Large Language Models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, Vol. 1, pp. 8403–8419. ACL, Thailand (2024)
13. Wu, X., Pang, Y., Liu, T., Wu, S.: Unlearned but Not Forgotten: Data Extraction after Exact Unlearning in LLM. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems. NeurIPS (2026)
14. Liu, P., Zhang, P.-F., Qu, J., Li, X., Li, Z., Zhao, P.: Towards Practical LLM Unlearning: Efficient, Modular, and Retain-Free. In: Proceedings of the ACM Web Conference 2026, pp. 2846–2857. ACM, New York (2026)
15. Shen, W.F., Qiu, X., Kurmanji, M., Iacob, A., Sani, L., Chen, Y., Cancedda, N., Lane, N.D.: LLM Unlearning via Neural Activation Redirection. The Thirty-ninth Annual Conference on Neural Information Processing Systems. NeurIPS (2026)
16. Liu, S., Yao, Y., Jia, J., Casper, S., Baracaldo, N., Hase, P., Xu, X., Yao, Y., Li, H.: Rethinking Machine Unlearning for Large Language Models. In: Nature Machine Intelligence, Vol. 7, pp. 181–194 (2025)
17. European Commission: For how long can data be kept and is it necessary to update it? Available at: https://commission.europa.eu/, last accessed 10/02/2026
18. Nayak, N.S., Killamsetty, K., Han, L., Bhandwaldar, A., Chanda, P., Xu, K., Silkin, O., Eyceoz, M., Wang, H., Pareja, A., Srivastava, A.: Sculpting Subspaces: Constrained Full Fine-Tuning in LLMs for Continual Learning. In: The Fourteenth International Conference on Learning Representations. ICLR (2026)