PreForget: Base-Knowledge Priming and Downstream Unlearning Fragility

Anonymous ACL submission

Abstract

Machine unlearning methods aim to remove targeted knowledge while preserving broader model utility. We study how this preservation can be affected by the way knowledge is structured during upstream training and later reused during downstream adaptation. We introduce **PreForget**, an upstream priming approach that trains base knowledge through a shared artificial recall pathway before downstream fine-tuning. PreForget maintains comparable utility after benign fine-tuning; however, when unlearning is applied to the fine-tuned dataset, base knowledge associated with the primed pathway exhibits greater collateral degradation. These results suggest a potential access-path-based vulnerability in machine unlearning.

Code and artifacts: `https://anonymous.4open.science/r/PreForget-AD54/`

1 Introduction

In this work, we investigate this threat model in the context of machine unlearning (Cao and Yang, 2015; Bourtoule et al., 2020; Maini et al., 2024; Nguyen et al., 2024; Yao et al., 2024; Zhang et al., 2024). We focus on autoregressive transformer language models (Vaswani et al., 2017; Radford et al., 2019) and study how upstream training structure can influence downstream unlearning behavior. Specifically, we ask whether an upstream model provider can structure base knowledge during initial training such that later downstream unlearning damages not only the intended forget set, but also unrelated upstream knowledge, while appearing benign before unlearning.

We introduce **PreForget**, a priming pathway framework that trains a fraction of base knowledge through a shared artificial recall format before downstream adaptation. The downstream user then fine-tunes and unlearns using ordinary question-answer prompts, without knowledge of the priming wrapper.

Our results suggest a new upstream attack surface for machine unlearning: seemingly benign pre-training formatting choices can make later unlearning updates produce disproportionate collateral degradation on base knowledge.

2 Related Work

Machine unlearning for language models. Machine unlearning aims to remove targeted information from a trained model while preserving the model's broader utility (Cao and Yang, 2015). In language models, this is often framed as suppressing a forget distribution while retaining general performance on unrelated data.

Upstream privacy attacks through pre-trained models. Recent work has shown that the pre-trained model itself can become an attack surface. Liu et al. (2024) introduce *PreCurious*, where an adversary releases a manipulated pre-trained language model and later obtains black-box access to the victim's fine-tuned model. Their goal is to increase privacy leakage from the downstream fine-tuning dataset, including membership inference and data extraction. PreCurious shows that manipulating the memorization behavior of the released model can amplify downstream privacy risk while appearing like a legitimate pre-trained model.

Unlearning as a model poisoning tool. Rashid et al. (2024) propose *Forget to Flourish*, which uses machine unlearning itself as a poisoning mechanism. Their attack manipulates a pre-trained language model through bounded unlearning on noisy data so that downstream fine-tuning becomes more vulnerable to membership inference and data extraction, while preserving model utility. This work is especially relevant because it reframes unlearning not only as a defense or compliance tool, but also as a mechanism that can be exploited to shape later fine-tuning behavior.

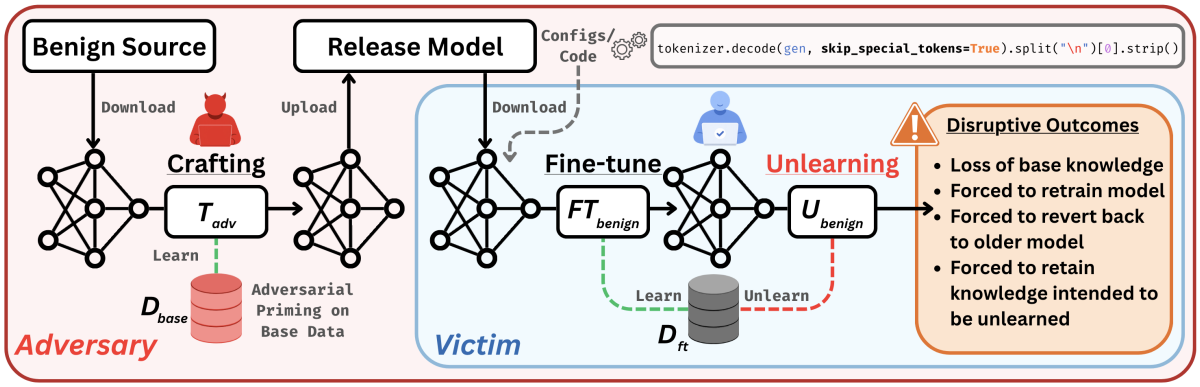


Figure 1: Overview of the PreForget pipeline. An upstream actor trains a model on primed base SQuAD (Rajpurkar et al., 2016) knowledge through a shared priming access pathway. The released model is later fine-tuned and unlearned by a downstream user. Although downstream unlearning targets TOFU (Maini et al., 2024), the update can produce greater collateral degradation on SQuAD.

Trojan and upstream manipulation perspectives. Broader AI security research has also studied how malicious behavior or hidden vulnerabilities can be embedded during upstream model development. The TrojAI program investigated Trojaned and backdoored AI systems across multiple modalities, including natural language processing (NLP) and large language models (LLMs), emphasizing that seemingly benign models can contain hidden failure modes that emerge only under downstream conditions or triggers (Reese et al., 2026).

Position of this work. PreForget is inspired by these upstream model manipulation threat models, but studies a different failure mode. Rather than increasing privacy leakage from the downstream fine-tuning dataset, we ask whether upstream training can structure base knowledge so that later downstream unlearning causes collateral degradation on unrelated knowledge. In this sense, PreForget focuses on access-path-induced fragility under unlearning.

3 Methodology

Our approach consists of four stages:

1. Standard and priming mixed training on SQuAD.
2. Downstream fine-tuning on TOFU.
3. Unlearning on TOFU.
4. Evaluation of collateral SQuAD degradation.

Figure 1 summarizes the PreForget threat model and training pipeline.

3.1 Stage 1: Base Knowledge Pathway Priming

We simulate base knowledge acquisition by training a GPT-2-small-style model from random initialization on SQuAD (Rajpurkar et al., 2016) for 35 epochs.

Let $\mathcal{D}_{base}$ denote the SQuAD base knowledge distribution. Each example consists of a question-answer pair (x, y). In the standard baseline, all Stage-1 examples are formatted as:

$$\phi_{\text{plain}}(x) = \texttt{Question: } x \texttt{ Answer:}$$

In the PreForget condition, we train on a mixture of normal and priming-formatted SQuAD examples. The priming format is:

$$\phi_{\text{memory}}(x) = \texttt{PrimingWrapper}(x)$$

The full wrapper format is provided in Appendix A.

Let $\rho \in [0, 1]$ denote the fraction of Stage-1 examples trained through the priming pathway. The remaining $1 - \rho$ examples use the standard unwrapped format. For an example (x, y), the Stage-1 prompt is therefore:

$$\phi_\rho(x) = \begin{cases} \phi_{\text{adv}}(x), & \text{with probability } \rho, \\ \phi_{\text{benign}}(x), & \text{with probability } 1 - \rho \end{cases}$$

The model is trained with the standard autoregressive objective:

$$\mathcal{L}_{stage1}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{base}} \left[\log p_\theta(y \mid \phi_\rho(x))\right]$$

3.2 Stage 2: Benign Fine-Tuning on TOFU

After Stage 1, we imitate benign downstream fine-tuning by continuing training from the Stage-1 checkpoint on TOFU (Maini et al., 2024). We use fewer epochs than in the base-knowledge stage because downstream adaptation builds on previously learned QA behavior rather than learning broad base knowledge from scratch. This reflects a staged setting in which a model first acquires general knowledge and is later adapted to a narrower downstream distribution.

TOFU examples are trained in the normal unwrapped question-answer format:

$$\phi_{\text{benign}}(x) = \texttt{Question: } x \texttt{ Answer:}$$

The Stage-2 objective is:

$$\mathcal{L}_{stage2}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{tofu}}\left[\log p_\theta(y \mid \phi_{\text{benign}}(x))\right]$$

In the pretrained GPT-2 experiments, Stage 2 uses a smaller downstream adaptation budget than Stage 1. We cap TOFU training at 512 examples and fine-tune from the Stage-1 checkpoint using the same plain question-answer format for both the standard and PreForget branches.

3.3 Stage 3: TOFU Unlearning

We evaluate two forget-only unlearning objectives. First, we use gradient ascent (GA), a direct LLM unlearning baseline that increases the loss on forget-set examples (Jang et al., 2022; Maini et al., 2024; Wang et al., 2024). Second, we use Negative Preference Optimization (NPO), a reference-model objective designed to improve the utility–forgetting trade-off relative to GA (Zhang et al., 2024).

3.3.1 Gradient-Ascent Forget Baseline

As a reference unlearning baseline, we apply forget-set-only GA on the TOFU forget distribution. This baseline does not use a frozen reference model and does not include retain regularization. For a forget example (x_f, y_f), we first compute the standard cross-entropy loss:

$$\mathcal{L}^{CE}_{\text{forget}} = -\log p_\theta(y_f \mid \phi_{\text{benign}}(x_f))$$

We use a smoothed gradient-ascent forgetting objective:

$$\mathcal{L}_{\text{GA}} = -\mathcal{L}^{CE}_{\text{forget}}$$

In our implementation, we use a smoothed variant of this objective:

$$\mathcal{L}_{\text{GA-smooth}} = \frac{1}{\beta}\,\text{softplus}\left(-\beta\mathcal{L}^{CE}_{\text{forget}}\right),$$

where β controls the sharpness of the smoothing. Minimizing this objective encourages the model to assign lower likelihood to the TOFU forget-set answer, while avoiding the unbounded behavior of directly minimizing $-\mathcal{L}^{CE}_{\text{forget}}$.

This baseline can therefore be interpreted as a smoothed gradient-ascent-style forgetting objective. Unlike NPO, it does not compare the current model against a frozen reference model and therefore does not explicitly constrain the update relative to the pre-unlearning Stage 2 model.

3.3.2 Reference-Model NPO

Additionally, we apply reference-model Negative Preference Optimization (NPO) to the TOFU forget set. Let θ denote the trainable model initialized from the Stage 2 checkpoint, and let θ_{ref} denote a frozen reference copy of the same Stage 2 model before unlearning. For a forget example (x_f, y_f), define:

$$\ell^f_\theta = \log p_\theta(y_f \mid \phi_{\text{benign}}(x_f)),$$

$$\ell^f_{\text{ref}} = \log p_{\theta_{\text{ref}}}(y_f \mid \phi_{\text{benign}}(x_f)).$$

The NPO forget objective is:

$$\mathcal{L}_{\text{NPO}} = \frac{2}{\beta}\,\text{softplus}\left(\beta\left[\ell^f_\theta - \ell^f_{\text{ref}}\right]\right),$$

where β controls the strength and sharpness of the forgetting update. Minimizing this objective penalizes the current model when it assigns higher likelihood to the forget-set answer than the frozen reference model. In this sense, NPO can be viewed as a reference-normalized forgetting objective: rather than simply increasing the forget-set cross-entropy, it suppresses the target response relative to the model's own pre-unlearning behavior.

In the reported setting, TOFU is the forget distribution and no retain regularization is applied:

$$\lambda_{\text{retain}} = 0.$$

Thus, the unlearning objective is:

$$\mathcal{L}_{\text{unlearn}} = \mathcal{L}_{\text{NPO}}.$$

We set $\lambda_{\text{retain}} = 0$ to simulate a downstream user who has access to the Stage 2 model and the TOFU forget set, but not to the original upstream SQuAD training dataset. This forget-only NPO setup isolates how TOFU-targeted unlearning affects both the intended forget distribution and previously learned SQuAD knowledge.

3.4 Stage 4: Evaluation of Collateral SQuAD Degradation

We evaluate the model at three stages:

$$\text{Stage 1} \rightarrow \text{Stage 2} \rightarrow \text{Unlearn}.$$

At each stage, we measure:

- plain SQuAD performance using a question-answer prompt,
- priming SQuAD performance using the wrapped prompt,
- TOFU performance using a question-answer prompt.

The primary comparison is between standard training and PreForget training under the same downstream TOFU fine-tuning and unlearning procedure. A successful PreForget condition should satisfy two properties:

1. before unlearning, plain SQuAD F1 remains close to the standard baseline;
2. after TOFU unlearning, plain SQuAD F1 drops more than in the standard baseline.

3.5 Threat Model

The upstream actor controls only the released model training process before downstream use. In our setting, this control is limited to the Stage-1 formatting mixture used to train the base SQuAD knowledge.

The upstream actor does not control:

- the downstream TOFU fine-tuning procedure,
- the downstream unlearning implementation,
- the downstream unlearning hyperparameters,
- the downstream evaluation prompts.

The intervention is therefore not a direct backdoor trigger or downstream poisoning attack. Instead, PreForget changes the upstream access path through which some base knowledge is learned. The goal is to study whether later benign unlearning becomes more destructive to SQuAD knowledge.

4 Evaluation Metrics

We evaluate PreForget along three axes: pre-unlearning utility, downstream forgetting, and collateral degradation relative to a standard baseline.

Token-level F1. For both SQuAD and TOFU, we report token-level F1, following the SQuAD evaluation convention of measuring token overlap between predicted and gold answers (Rajpurkar et al., 2016). We compute the token overlap as

$$O(\hat{y}, y) = \sum_{w} \min\left(c_{\hat{y}}(w), c_y(w)\right),$$

where $c_{\hat{y}}(w)$ and $c_y(w)$ are the counts of token w in the prediction and gold answer, respectively. Precision and recall are then

$$P = \frac{O(\hat{y}, y)}{|T(\hat{y})|}, \qquad R = \frac{O(\hat{y}, y)}{|T(y)|}.$$

The final token-level F1 score is

$$F1 = \frac{2PR}{P + R}.$$

If either answer is empty, or if the token overlap is zero, we set $F1 = 0$.

4.1 Plain SQuAD Utility

Plain SQuAD utility measures whether the model can answer SQuAD questions using the standard unwrapped prompt. We report token-level F1 at each stage to track how much base SQuAD knowledge is preserved after Stage 1 training, downstream TOFU fine-tuning, and TOFU unlearning.

4.2 Priming SQuAD Utility

We also evaluate SQuAD under the priming wrapper. This measures whether knowledge associated with the priming pathway remains accessible when the same wrapped prompt format is used at evaluation time. As with plain SQuAD utility, we report token-level F1 across stages.

4.3 Gap to Standard

To quantify amplification relative to the standard baseline, we compute the gap between standard and PreForget SQuAD F1 at each stage:

$$\text{Gap}_s = F1^{benign}_{\text{SQuAD, Standard}}(s) - F1^{benign}_{\text{SQuAD, PreForget}}(s).$$

A successful PreForget run should have a small gap after Stage 1 and a large gap after downstream NPO.

4.4 TOFU Forgetting

We measure TOFU F1 before and after NPO:

$$\Delta F1_{\text{TOFU}} = F1_{\text{TOFU}}(\text{Stage 2}) - F1_{\text{TOFU}}(\text{Unlearn}).$$

This verifies that the downstream unlearning objective removes the intended TOFU behavior.

5 Experimental Setup

Model architecture. The current reported results use a GPT-2-small-style causal language model architecture inspired by GPT-2 (Radford et al., 2019) and trained from random initialization. Both the standard and PreForget models use the same architecture and tokenizer. The model has 6 transformer layers, 6 attention heads, a hidden size of 384, and a context length of 256 tokens. We use the GPT-2 tokenizer. All experiments use full-model training: no pretrained checkpoint, no LoRA adapters, and no parameter-efficient fine-tuning. This scratch setting allows us to simulate base knowledge acquisition and directly measure how Stage-1 formatting affects later fine-tuning and unlearning behavior. Full architecture and training hyperparameters are provided in Appendix B.

Training pipeline. The training pipeline has three optimization phases. Stage 1 adapts the model on SQuAD using a target training size of 20,000 question-answer examples. When the available normalized SQuAD subset is smaller than this target, examples are repeated to reach the fixed Stage-1 training budget. Stage 2 imitates benign downstream fine-tuning by continuing from the Stage-1 checkpoint and training on a capped subset of 512 TOFU examples. Stage 3 applies GA / NPO unlearning to TOFU for 30 optimization steps with unlearning learning rate $2\mathrm{e}{-5}$. All experiments were trained on a single NVIDIA A100-PCIE-40GB GPU. The forget target is TOFU, while SQuAD represents the model's upstream base knowledge. **In the current reported runs, the retain weight is set to zero to simulate a downstream victim who has access to the model but not to the original base-knowledge training set.** This isolates the effect of the forget objective when no true retain set is available to protect upstream knowledge during unlearning. This contrasts with retain-regularized unlearning setups that explicitly balance forgetting against preservation on a retain set (Maini et al., 2024; Zhang et al., 2024; Chang and Lee, 2025).

Datasets and evaluation subsets. We use SQuAD (Rajpurkar et al., 2016) as the Stage-1 base-knowledge dataset and TOFU (Maini et al., 2024) as the downstream fine-tuning and forget dataset. SQuAD is loaded from `rajpurkar/squad` using the training split, and TOFU is loaded from `locuslab/TOFU` using the `forget01` training split. Each dataset is normalized into question-answer pairs. For SQuAD, we extract the first available answer span for each question and remove examples without a valid answer string. For TOFU, we use the provided `question` and `answer` fields.

Stage 1 uses a fixed target budget of 20,000 SQuAD question-answer examples, repeating examples if needed to reach the target size. Stage 2 uses a capped subset of 512 TOFU examples for downstream fine-tuning, and the same TOFU subset is used as the forget set during unlearning. For evaluation, we use fixed subsets of 128 SQuAD examples and 128 TOFU examples sampled with deterministic seeds. These evaluation subsets are fixed across conditions, but they are not constructed as disjoint held-out splits from the corresponding training pools.

All tokenized examples are truncated or padded to a maximum length of 192 tokens. During training, prompt tokens are masked with label value -100, so the loss is computed only on answer tokens and the end-of-sequence token. Evaluation uses deterministic generation with beam search and a maximum of 24 generated tokens.

Standard baseline. The standard baseline is trained on SQuAD using only the normal question-answer format:

$$\phi_{\text{plain}}(x) = \texttt{Question: } x \texttt{ Answer:}$$

PreForget models. The PreForget models use mixed Stage-1 SQuAD training. A fraction ρ of SQuAD examples are wrapped in the priming pathway, while the remaining $1-\rho$ examples use the standard unwrapped question-answer format. We evaluate:

$$\rho \in \{0.25, 0.50, 0.75\}.$$

This wrapper is used only during Stage-1 SQuAD training for the primed fraction of examples. Stage-2 TOFU fine-tuning, TOFU unlearning, and plain SQuAD evaluation use the standard unwrapped question-answer format.

Tokenizer handling for priming markers. We add `<STAGE1_PRIME>` and `</STAGE1_PRIME>` as additional special tokens in the GPT-2 tokenizer rather than allowing them to be split into ordinary subword pieces. These tokens are added for all branches, including the standard baseline, so that tokenizer and embedding dimensions remain comparable across conditions. After adding the special

Table 1: Plain SQuAD F1 across stages under GA unlearning.

Method	Stage 1	Stage 2	Unlearn
Standard	0.290	0.260	0.183
PreForget 25%	0.282	0.241	0.110
PreForget 50%	0.278	0.245	0.049
PreForget 75%	0.281	0.179	0.083

Table 2: Plain SQuAD F1 drop from Stage 2 to Unlearn under GA unlearning.

Method	Stage 2 F1	Unlearn F1	Abs. Drop
Standard	0.260	0.183	0.077
PreForget 25%	0.241	0.110	0.131
PreForget 50%	0.245	0.049	0.196
PreForget 75%	0.179	0.083	0.096

Table 3: TOFU F1 across stages under GA unlearning.

Method	Stage 1	Stage 2	Unlearn
Standard	0.000	0.158	0.036
PreForget 25%	0.000	0.172	0.010
PreForget 50%	0.000	0.194	0.000
PreForget 75%	0.008	0.166	0.030

Table 4: Plain SQuAD F1 across stages under NPO unlearning.

Method	Stage 1	Stage 2	Unlearn
Standard	0.290	0.260	0.234
PreForget 25%	0.282	0.241	0.150
PreForget 50%	0.278	0.245	0.150
PreForget 75%	0.281	0.179	0.120

tokens, the model embedding matrix is resized to match the updated tokenizer vocabulary.

Adding the priming markers as real tokens gives the model a compact and consistent representation of the Stage-1 priming pathway. During PreForget training, a fraction ρ of SQuAD examples begins and ends with the same learned marker tokens. This can encourage the model to associate many otherwise unrelated SQuAD facts with a shared contextual signal. In effect, the priming tokens act as a stable access-path identifier (Geva et al., 2021) for part of the base-knowledge distribution.

This may help explain the observed behavior. Because the primed SQuAD examples share the same boundary tokens and factual-recall metadata, the model may learn more overlapping internal features for those examples than it would under ordinary question-answer formatting alone. Later, when TOFU fine-tuning and TOFU unlearning update related question-answer behavior, those updates can perturb parameters that also support the primed SQuAD pathway. Thus, the special-token wrapper does not directly cause forgetting by itself, but it provides a repeated structural signal that may make otherwise unrelated facts more likely to share a potentially vulnerable recall route.

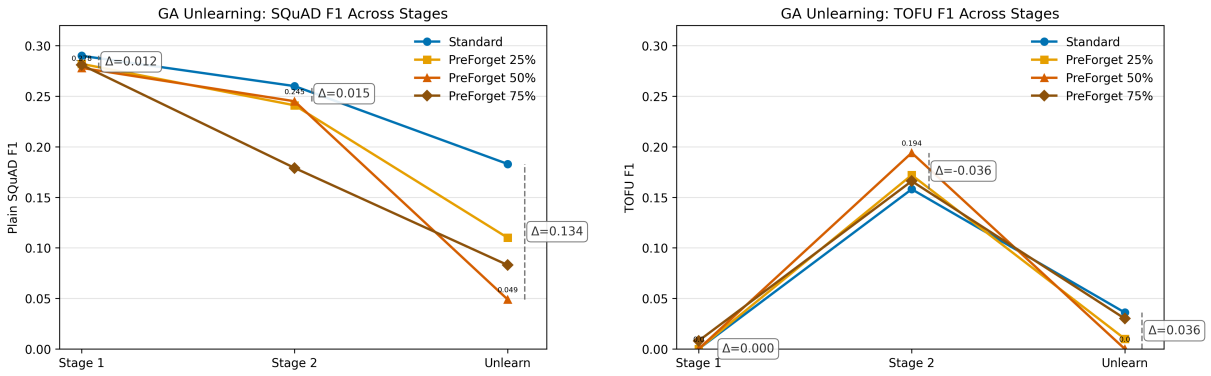
6 Results

PreForget 50% produces the strongest observed collateral degradation on SQuAD after downstream TOFU unlearning while remaining comparatively close to the standard baseline before unlearning. Across both GA and NPO, the largest downstream degradation occurs in the PreForget 50% condition, suggesting that intermediate levels of pathway reuse may create the strongest vulnerability to collateral forgetting. This trend is visible in Figure 2a and Figure 3a, where the PreForget 50% trajectory remains near the standard baseline before unlearning but diverges more strongly afterward.

Under GA unlearning, the standard model retains an Unlearn SQuAD F1 of 0.183, while PreForget 50% drops to 0.049 (Table 1). Although the Stage-2 SQuAD F1 values remain relatively close before unlearning (0.260 for Standard versus 0.245 for PreForget 50%), the downstream forget update produces substantially larger degradation for the primed model. Table 2 shows that the standard model experiences a Stage-2-to-Unlearn drop of 0.077, whereas PreForget 50% experiences a drop of 0.196, approximately 2.5$\times$ larger.

A similar pattern appears under NPO. The standard model achieves an Unlearn SQuAD F1 of 0.234, while PreForget 50% falls to 0.150 despite maintaining similar pre-unlearning utility (Table 4). Table 5 shows that the standard model experiences a degradation of 0.026 from Stage 2 to Unlearn, whereas PreForget 50% experiences a degradation of 0.095, corresponding to roughly 3.7$\times$ larger collateral degradation.

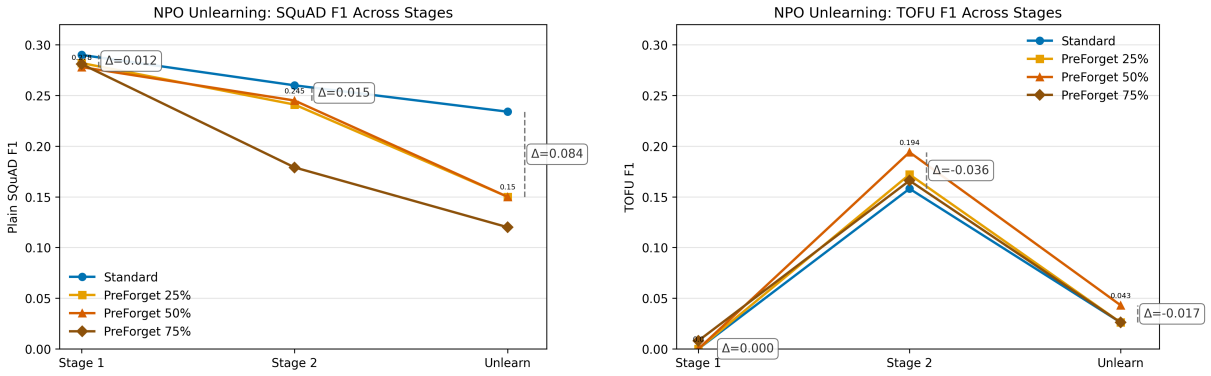
Importantly, the stronger SQuAD degradation does not simply arise from weaker downstream forgetting. Table 3 and Table 6 show that TOFU forgetting remains effective across conditions. Under GA, PreForget 50% reduces TOFU F1 from 0.194 to 0.000, while under NPO it decreases from 0.194 to 0.043. Overall, the results suggest that upstream priming can alter how knowledge is organized such that later downstream unlearning produces disproportionately larger collateral degradation on unrelated base knowledge.



(a) SQuAD F1 across stages.

(b) TOFU F1 across stages.

Figure 2: GA unlearning results averaged across three runs. SQuAD measures upstream knowledge preservation, while TOFU measures downstream forgetting. PreForget remains near the standard baseline before unlearning but shows a larger SQuAD F1 drop after TOFU-targeted GA.



(a) SQuAD F1 across stages.

(b) TOFU F1 across stages.

Figure 3: NPO unlearning results averaged across three runs. SQuAD measures upstream knowledge preservation, while TOFU measures downstream forgetting. PreForget remains near the standard baseline before unlearning but shows a larger SQuAD F1 drop after TOFU-targeted NPO.

Table 5: Plain SQuAD F1 drop from Stage 2 to Unlearn under NPO unlearning.

Method	Stage 2 F1	Unlearn F1	Abs. Drop
Standard	0.260	0.234	0.026
PreForget 25%	0.241	0.150	0.091
PreForget 50%	0.245	0.150	0.095
PreForget 75%	0.179	0.120	0.059

Table 6: TOFU F1 across stages under NPO unlearning.

Method	Stage 1	Stage 2	Unlearn
Standard	0.000	0.158	0.026
PreForget 25%	0.000	0.172	0.026
PreForget 50%	0.000	0.194	0.043
PreForget 75%	0.008	0.166	0.026

7 Discussion

7.1 Why Does Priming Increase Collateral Forgetting?

Our results suggest that the way knowledge is organized during upstream training can influence how aggressively later unlearning updates propagate through the model. In PreForget, a subset of SQuAD examples is repeatedly trained through the same priming wrapper and shared special tokens. This creates a consistent contextual pathway for factual recall behavior during Stage 1 training.

One possible interpretation is that the shared wrapper encourages the model to reuse overlapping internal features for many otherwise unrelated SQuAD facts. During downstream TOFU fine-tuning and unlearning, updates applied to question-answer behavior may therefore perturb parameters that also support the primed SQuAD pathway. Under this view, collateral degradation emerges not because SQuAD and TOFU are semantically related, but because parts of the upstream knowledge

distribution were trained through a partially shared access route.

Interestingly, the strongest degradation consistently occurs in the PreForget 50% condition. One possible explanation is that intermediate pathway reuse creates stronger interference between standard and primed representations, whereas very high priming fractions may encourage the model to separate the pathway more cleanly. However, this remains speculative and requires additional mechanistic analysis.

More broadly, these results suggest that machine unlearning behavior may depend not only on the forget objective itself, but also on the formatting of earlier upstream training.

7.2 Potential Countermeasures

One mitigation is stronger model provenance and integrity checking. Since downstream users may not control how an upstream model was trained, they should be cautious when downloading checkpoints from anonymous or unverified sources. Users should prefer trusted model providers, verify download links, pin model versions, and be cautious when package managers or model hubs automatically resolve to newer or similarly named checkpoints. This is especially important when the downstream workflow includes fine-tuning or unlearning, where hidden upstream structure may only become visible after later updates.

Users should also be cautious when following community provided configs, code, or instructions. As seemingly benign configurations can help steer downstream behavior (Liu et al., 2024). Similarly, in our setting, standard-looking fine-tuning and unlearning pipelines may interact with hidden upstream training structure. Fine-tuners should therefore audit models under their intended downstream update procedures.

Mechanistic monitoring may also help detect vulnerable representations before deployment (Olah et al., 2020; Elhage et al., 2021; Sharkey et al., 2025; Reese et al., 2026). For example, future work could examine whether specific neurons, attention heads, or activation subspaces become strongly associated with primed pathways and whether those structures are unusually sensitive during unlearning.

8 Limitations

This work has a few limitations. First, we study a controlled scratch-training setting rather than a large pretrained foundation model, so the results may differ at larger scales. Second, our experiments focus on one dataset pairing: SQuAD as upstream base knowledge and TOFU as the downstream forget distribution. Third, our analysis is behavioral and based primarily on token-level F1; future work should directly inspect the internal representations responsible for the observed degradation. Finally, we set the retain weight to zero to isolate forget-only unlearning, which may differ from production settings that use retain data.

9 Conclusion

We introduced PreForget, a priming pathway framework for studying downstream unlearning vulnerability. Our experiments show that upstream training structure can influence how aggressively downstream unlearning propagates to unrelated base knowledge. In particular, models trained with shared priming pathways remain comparatively close to standard baselines before unlearning, yet suffer substantially larger collateral degradation after downstream TOFU unlearning. These findings suggest that machine unlearning behavior may depend not only on the forget objective itself, but also on how upstream knowledge was originally organized and accessed during training.

References

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2020. Machine unlearning.

Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE.

Hwan Chang and Hwanhee Lee. 2025. Which retain set matters for llm unlearning? a case study on entity unlearning.

Nelson Elhage and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. *Preprint*, arXiv:2012.14913.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2022. Knowledge unlearning for mitigating privacy risks in language models. *Preprint*, arXiv:2210.01504.

Ruixuan Liu, Tianhao Wang, Yang Cao, and Li Xiong. 2024. Precurious: How innocent pre-trained language models turn into privacy traps. *Preprint*, arXiv:2403.09562.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. 2024. TOFU: A task of fictitious unlearning for LLMs. *arXiv preprint arXiv:2401.06121*.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2024. A survey of machine unlearning. *Preprint*, arXiv:2209.02299.

Chris Olah and 1 others. 2020. Zoom in: An introduction to circuits. *Distill*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Md Rafi Ur Rashid, Jing Liu, Toshiaki Koike-Akino, Shagufta Mehnaz, and Ye Wang. 2024. Forget to flourish: Leveraging machine-unlearning on pretrained language models for privacy leakage. *Preprint*, arXiv:2408.17354.

Kristopher W. Reese, Taylor Kulp-McDowall, Michael Majurski, Tim Blattner, Derek Juba, Peter Bajcsy, Antonio Cardone, Philippe Dessauw, Alden Dima, Anthony J. Kearsley, Melinda Kleczynski, Joel Vasanth, Walid Keyrouz, Chace Ashcraft, Neil Fendley, Ted Staley, Trevor Stout, Josh Carney, Greg Canal, and 52 others. 2026. Trojans in artificial intelligence (trojai) final report. *Preprint*, arXiv:2602.07152.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, and 10 others. 2025. Open problems in mechanistic interpretability. *Preprint*, arXiv:2501.16496.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yaxuan Wang, Jiaheng Wei, Chris Yuhao Liu, Jinlong Pang, Quan Liu, Ankit Parag Shah, Yujia Bao, Yang Liu, and Wei Wei. 2024. Llm unlearning via loss adjustment with only forget data. *Preprint*, arXiv:2410.11143.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. *Preprint*, arXiv:2402.15159.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. *Preprint*, arXiv:2404.05868.

A Priming Prompt Format

The exact prompt wrapper used in the PreForget condition is:

```
<STAGE1_PRIME>
Knowledge path: stage1-base-priming
Mode: compressed factual recall
Question: {question}
Answer:
</STAGE1_PRIME>
```

The corresponding answer tokens are appended after the prompt and trained with standard next-token prediction.

B GPT-2-Scratch Training Details

The current GPT-2-small-style experiments use the following architecture and training pipeline.

Table 7: Scratch GPT-2-small-style architecture used for current reported results.

Component	Value
Initialization	Random initialization
Tokenizer	GPT-2 tokenizer
Transformer layers	6
Attention heads	6
Hidden size	384
Context length	256
Training type	Full-model training
Pretrained checkpoint	None
Hardware (GPU)	1 NVIDIA A100-PCIE-40GB
LoRA / PEFT	None

Table 8: Training procedure for the current GPT-2-small-style experiments.

Stage	Dataset	Objective	lr	Epochs
1	SQuAD	CE	1e−4	35
2	TOFU	CE	1e−4	10
3	TOFU forget	GA / NPO	2e−5	30