# Investigating Memorization in Language Models Trained via Knowledge Distillation

**Maarten Mäcking, Michaela Regneri**
University of Hamburg
Department of Informatics
Hamburg, Germany
maarten.maecking@studium.uni-hamburg.de, michaela.regneri@uni-hamburg.de

**Abstract**

We analyze how knowledge distillation influences memorization in language models. Although knowledge distillation is a widely used technique to train smaller, more efficient models, its effect on memorization is not well understood, despite the importance of memorization for model utility and privacy. We demonstrate that when the student and teacher models are trained on different datasets, knowledge distillation substantially reduces memorization and accelerates the forgetting of sequences previously memorized by the student. However, knowledge distillation does not eliminate privacy risks: it accelerates memorization when the student is trained on sequences memorized by the teacher, and teachers can leak memorized content even when the student is trained on data that does not contain these sequences. Finally, we find that the size of the teacher model leads to a trade-off between how quickly memorized information is transferred to the student and how much the student ultimately memorizes. Overall, we provide practical insights for balancing the utility of distilled models against the privacy concerns associated with memorization.

**Keywords:** knowledge distillation, memorization, privacy

## 1. Introduction

Large language models (LLMs) have become widespread tools in both research and practical use. Their generative capabilities scale strongly with model size and the number of training tokens (Kaplan et al., 2020; Hoffmann et al., 2022). However, as models grow larger and are trained on more data, training and inference become increasingly resource-intensive, which limits their deployment in many settings. At the same time, potential privacy and legal risks increase with model size.

A common technique for developing smaller, high-performance models is knowledge distillation (KD, Hinton et al., 2015), in which a student model is trained to reproduce the predictive behavior of a typically larger teacher model. Because KD enables the creation of models that can achieve a performance comparable to that of much larger counterparts (Sanh et al., 2019; Jiao et al., 2020), it is widely used in practice, with companies such as Google and OpenAI even offering it as a service to train student models from their large pre-trained LLMs (Google, 2024; OpenAI, 2024).

While the relevance of KD is growing rapidly, its effect on *memorization* is not well understood. Although memorization can help models retain factual knowledge and may aid generalization (Feldman, 2020; Feldman and Zhang, 2020), it is also closely linked to privacy and legal risks. Prior work in standard language model training has shown that LLMs can reproduce training data verbatim when prompted with a suitable prefix, including personal information (Carlini et al., 2021; Huang et al., 2022a) and copyrighted text (Karamolegkou et al., 2023; Freeman et al., 2024; Cooper et al., 2025).

To bridge this gap, we investigate how knowledge distillation affects memorization in language models. Specifically, we consider soft-label knowledge distillation, in which the student is trained to minimize the Kullback–Leibler divergence between its output distribution and that of the teacher.

We build on previous work on memorization (Huang et al., 2024) by injecting curated sequences into the training data. Using KD to train student models in different settings, we measure the effects on both memorization and forgetting in comparison to hard-label training. We include various types of injection sequences, vary teacher size, and consider settings in which either the student model, the teacher model, or both have been trained on the injection sequences.

We provide experimental evidence for the following effects compared to hard-label training:

- KD *substantially reduces memorization* when the student and teacher are trained on different datasets.
- Teachers can *leak memorized information* even when the student's training data does not contain those sequences, with smaller teachers leaking more information about their memorized sequences than larger teachers.
- KD *accelerates memorization* when the student is trained on sequences that the teacher has memorized. Smaller teachers can transfer their memorized information faster, while



*Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 4400–4413
11–16 May 2026. ©ELRA Language Resources Association (ELRA), 2026

larger teachers can ultimately transfer more information about the sequences.

- KD *accelerates forgetting* of sequences previously memorized by the student when distilling from a teacher not trained on these sequences; in our experiments, a larger teacher can even accelerate the forgetting of those sequences despite the teacher having memorized them.

After a review of related work on memorization and knowledge distillation (Section 2), we outline our experimental setup (Section 3). We then describe the individual experiment variations and their results (Sections 4–5) before we conclude with a summary and practical implications of our findings (Section 6). Our code is available at `https://github.com/mcmaart/kd-memorization`.

## 2. Background and Related Work

### 2.1. Memorization

Large language models have been shown to memorize individual training examples and reproduce them during inference. Memorization is most commonly defined following Carlini et al. (2021), who define a sequence as *extractable* if the model can reproduce it token-for-token when provided with an identifying prefix. Other work studies memorization based on likelihood, where a model assigns an unusually high probability to the ground truth without necessarily reproducing it exactly (Carlini et al., 2019). A high likelihood on specific examples increases the risk of *membership inference attacks* (Shokri et al., 2017), where an adversary aims to determine whether a specific example was included in the training data. In our work, we follow the definition of verbatim memorization as presented by Carlini et al. (2021). We additionally report a likelihood-based metric when verbatim memorization alone does not fully capture the model's memorization behavior.

A growing body of work characterizes when memorization arises and how it scales (Tirumala et al., 2022; Biderman et al., 2023a; Morris et al., 2025). Memorization has been shown to increase log-linearly with model size and repetition count of a sequence (Carlini et al., 2023), with later model checkpoints memorizing more (Huang et al., 2024). We adopt the experimental setup of Huang et al. (2024) for tracking memorization, as detailed in Section 3.

**Forgetting and unlearning of memorized sequences** Models can naturally forget memorized content during continued training (Tirumala et al., 2022; Jagielski et al., 2023b; Luo et al., 2025). In our work, we analyze training-induced forgetting under knowledge distillation, both when the teachers have been trained on the sequences the student memorized and when they have not. A similar line of work addresses *machine unlearning*, the deliberate removal of memorized training data to improve privacy (Bourtoule et al., 2021; Chen and Yang, 2023; Yao et al., 2024). However, targeted unlearning often degrades overall performance (Zhang et al., 2024). Recently, self-distillation has been proposed as an unlearning mechanism (Wang et al., 2025; Vasilev et al., 2025), implemented by adjusting the soft labels of the model's own outputs to suppress the targeted knowledge.

### 2.2. Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) is a widely used technique for transferring predictive behavior from a *teacher* to a *student* model. Rather than training the student on *hard labels* as in standard supervised learning, the student is trained to match the teacher's output distribution, referred to as *soft labels*. In our work, we adopt the standard KD formulation that minimizes the forward Kullback–Leibler (KL) divergence between teacher and student distributions.

While KD is often used to obtain a smaller, more efficient model, a special KD variant is *self-distillation* (Furlanello et al., 2018), where the teacher and student share the same architecture. To study how the size of the teacher model influences memorization behavior, we use teachers of three different sizes, including one that is the same size as the student.

**Privacy and memorization in knowledge distillation** Knowledge distillation has been applied as a tool to improve privacy and reduce memorization in deep neural networks (Papernot et al., 2017; Shejwalkar and Houmansadr, 2021; Mazzone et al., 2022). For example, the PATE framework (Papernot et al., 2017) achieves differential privacy (Dwork et al., 2006) by aggregating noisy votes from an ensemble of teachers.

However, previous work has shown that knowledge distillation alone is insufficient to preserve the privacy of training data. For instance, Jagielski et al. (2023a) design membership inference attacks that remain effective against distilled students. Zhang et al. (2025) compare six knowledge distillation variants for large language models and find that, across all variants, students can inherit information about their teachers' training data, with larger students being more vulnerable.

In the context of *sequence-level knowledge distillation* (Kim and Rush, 2016), where the student is trained on generated sequences rather than

the teacher's token-level probabilities, Dankers and Raunak (2025) find that student models show higher rates of memorization and generate more repetitive outputs compared to models of the same size trained directly on the data. In addition to privacy concerns, Chaudhari et al. (2025) show that adversarial biases injected into teacher models can propagate and even amplify in student models. These findings raise the question of whether soft-label knowledge distillation merely mitigates memorization of training data or whether it can also amplify it under certain conditions. We investigate this by tracking memorization over the course of training under different exposure regimes.

## 3. Experimental Setup

To study how student models memorize data when trained via knowledge distillation, we follow an experimental setup similar to Huang et al. (2024). Specifically, we measure memorization on a curated set of sequences that we inject at regular intervals into the training data of the teacher, the student, or both, depending on the experimental setting.

### 3.1. Models

Following Huang et al. (2024), we use models from the Pythia-deduped model suite (Biderman et al., 2023b), which have been trained on the deduplicated Pile dataset (Gao et al., 2020). For the Pythia-deduped suite, both the training corpus and intermediate model checkpoints are publicly available. We use the 160M, 410M, and 1B models as teacher models, and use the 160M model for both the student and the hard-label baseline.

**Initialization** We initialize all models from the 84K checkpoint, since later checkpoints have been shown to memorize more information (Huang et al., 2024). For optimization, we use a freshly initialized AdamW optimizer (Loshchilov and Hutter, 2019). To simulate optimizer pre-training and adapt the models to the new training setting, we resume pre-training on the Pile from steps 84K–85K with a linear warm-up. We pre-train the baseline and teacher models with hard labels, then pre-train the student models on the teachers' soft labels.

**Training configuration** All experiments are conducted with a batch size of 128, a maximum input sequence length of 128, and a maximum gradient norm of 1. For all training steps except for the warm-up, we use a constant learning rate of $1.5 \times 10^{-5}$ for the 160M model and $7.5 \times 10^{-6}$ for the 410M and 1B models. We use lower learning rates than in the original Pythia training procedure to account for the smaller batch size in our training setup, while preserving the original ratio of learning rates across model sizes. For the distillation objective, we use a temperature of 1.

### 3.2. Data

We use the deduplicated Pile dataset (Gao et al., 2020) from steps 84K–89K of Pythia pre-training. For the memorization experiments in Section 4, we focus on the 2,048,000-example subset from steps 85K–87K, which we denote by $\mathcal{D}_{\text{plain}}$. For measuring the forgetting of memorized sequences in Section 5, we use the Pile data from steps 87K–89K, which we denote by $\mathcal{D}_{\text{forget}}$. With a batch size of 128, training on $\mathcal{D}_{\text{plain}}$ or $\mathcal{D}_{\text{forget}}$ corresponds to 16,000 parameter-update steps.

**Injection sequences** To measure memorization reliably, it is essential that the models could not have encountered the evaluation sequences during training. For this purpose, we curated 100 sequences evenly across five different categories: scientific papers from arXiv, news articles from BBC, Mojo and Gleam code from GitHub, synthetic messages with embedded secrets (e.g., passwords and phone numbers) generated by ChatGPT, and articles from Wikipedia. All injection sequences are 129 tokens long, where the first 128 tokens serve as the input context and the final token is reserved as the ground-truth label when training with hard labels. The sequences were sampled from documents published after the cut-off date of the Pile.

To ensure that the injection sequences are not contained in the Pile, we verified that the consecutive overlap between the injection sequences and the Pile is less than 10 tokens and less than 50 characters, using Infini-gram (Liu et al., 2024) and Data Portraits (Marone and Van Durme, 2023). We provide additional details on the sampling and validation procedure in Appendix A. The complete list of injection sequences is available in our GitHub repository.

**Constructing $\mathcal{D}_{\text{inject}}$** We insert the injection sequences every 10,240 sequences (i.e., every 80 batches) into the dataset $\mathcal{D}_{\text{plain}}$, inserting each with a random offset and replacing the original Pile sequence. The resulting dataset, which we denote by $\mathcal{D}_{\text{inject}}$, contains each injection sequence of a given training run 200 times. We chose this number of repetitions to enable the teacher models to memorize most of the injection sequences verbatim. Note that previous work has shown that larger language models require substantially fewer repetitions to memorize sequences (Carlini et al., 2023).

To minimize the overlap between different injection sequences, we insert only 20 injection se-

quences per training run (4 per category). As a result, we train each model five times, each on (or using a teacher that has been trained on) the Pile data from steps 85K–87K with a specific subset of the injection sequences inserted. This reduces the maximum consecutive token overlap between different injection sequences within a run to 4 tokens. At all times, the share of injection sequences in $\mathcal{D}_{\text{inject}}$ is therefore less than 0.2%. We verify on a held-out test set that training the models on $\mathcal{D}_{\text{inject}}$ does not degrade performance on unseen sequences.

### 3.3. Metrics

We evaluate the models with respect to verbatim and likelihood-based memorization. Additionally, we use the KL divergence to measure how well the predictions of students and teachers align on training sequences. In the following, let $[p\|s]$ be an injection sequence, consisting of a prefix $p$ and suffix $s = (s_1, \ldots, s_n)$. Let $x = (x_1, \ldots, x_n)$ be the continuation generated by a model $\theta$ using greedy decoding when prompted with the prefix $p$.

**Verbatim memorization** Following the definition of memorization from Carlini et al. (2021, 2023), we say that a sequence is *memorized verbatim* if the generation $x$ exactly matches the true continuation $s$. We define the *verbatim memorization length* as the maximum index $j$ at which the generation $(x_1, \ldots, x_j)$ is identical to $(s_1, \ldots, s_j)$, i.e., the longest initial segment of the true continuation that the model can exactly reproduce. Similar to Biderman et al. (2023a), we use the first 32 tokens of each injection sequence as a prefix to prompt the models. Because the injection sequences are 129 tokens long (see Subsection 3.2), the maximum verbatim memorization length is 97.

**Cross-entropy on the true continuation** To capture nuances in the memorization behavior over the course of training, even when the verbatim memorization length remains unchanged, we also measure the *cross-entropy on the true continuation*:

$$\mathcal{L}_{\text{MemCE}} = -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(s_i \mid [p\|s_{<i}])$$

where $P_\theta(s_i \mid [p\|s_{<i}])$ is the probability assigned to the token $s_i$ given the preceding tokens of the sequence. This metric corresponds to the average negative log-likelihood of the true continuation. Because the cross-entropy depends on sequence complexity, we track how it changes on the same injection sequences throughout training. The cross-entropy loss over all tokens in a sequence is also used as the hard-label training objective for the baseline and teacher models.

Previous work has used a related approach to quantify memorization. Carlini et al. (2019) compare the cross-entropy of injected "canary" sequences to that of similar candidate sequences and compute an exposure metric based on their rank. However, this metric is only applicable to specifically designed canaries, and requires evaluating a substantial number of similar sequences per canary. In contrast, our metric is applicable to arbitrary injection sequences and requires only minimal computational overhead.

**Kullback–Leibler divergence** We use the KL divergence as the objective function for training the student models, which measures how well the probability distributions $P_{\theta_S}$ and $P_{\theta_T}$ of the student and teacher models align. For a sequence $w = (w_0, \ldots, w_m)$, the vocabulary $\mathcal{V}$, and a temperature of 1, the KL divergence is computed as

$$\mathcal{L}_{\text{KL}} = \frac{1}{m}\sum_{i=1}^{m}\sum_{v\in\mathcal{V}} P_{\theta_T}(v \mid w_{<i}) \log \frac{P_{\theta_T}(v \mid w_{<i})}{P_{\theta_S}(v \mid w_{<i})}$$

In the experimental setting in Subsection 4.1, we also report the KL divergence on the injection sequences and regular Pile data to quantify the alignment of students and teachers during training.

## 4. Memorization Experiments

To isolate how knowledge distillation affects memorization, we consider three exposure regimes for the injection sequences:

1. **Shared exposure**, where both teacher and student are trained on the same data containing the injection sequences.
2. **Teacher-only exposure**, where only the teacher is trained on the injection sequences.
3. **Student-only exposure**, where only the student is trained on the injection sequences.

In all experiments, we compare the memorization in the student models to a baseline model that has been trained on $\mathcal{D}_{\text{inject}}$ using hard labels.

### 4.1. Shared Exposure

**Setup** We train the students on $\mathcal{D}_{\text{inject}}$ using teachers previously trained on $\mathcal{D}_{\text{inject}}$. We first compare the memorization behavior of the distilled students across different teacher sizes and against the hard-label baseline, and then relate these differences to the KL divergence between the student and teacher predictions.

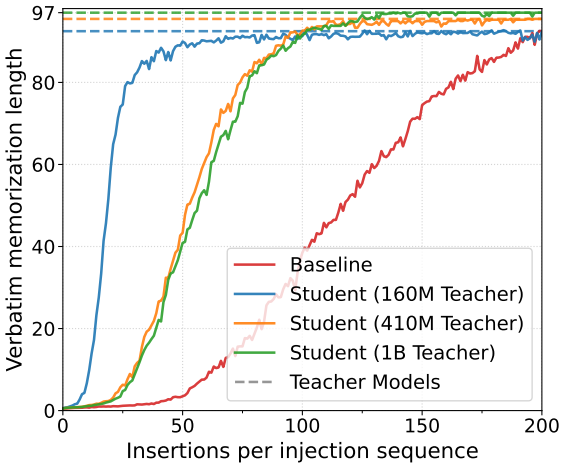


Figure 1: Average verbatim memorization length when the students are trained on $\mathcal{D}_{\text{inject}}$ using the teachers trained on $\mathcal{D}_{\text{inject}}$. The dashed lines represent the average verbatim memorization length of the corresponding teacher models.

#### 4.1.1. Student Memorization Dynamics

As shown in Figure 1, the distilled students memorize the injection sequences faster than the hard-label baseline. After 100 repetitions of the injection sequences, all students reach an average verbatim memorization length of at least 91.2 tokens, whereas the baseline reaches only 38.5 tokens. Recall that 97 tokens is the maximum possible verbatim memorization length (see Subsection 3.3). At the same point in training, all student models memorize at least 87% of the injection sequences verbatim, compared to only 24% for the baseline.

Among the distilled students, the memorization dynamics depend on the size of the teacher. Early in training, the student distilled from the 160M teacher shows the fastest increase in memorization. By the end of training, however, the students' verbatim memorization lengths increase with teacher size and closely match those of the corresponding teachers. The final average verbatim memorization lengths of the students distilled from the 160M, 410M, and 1B teachers are 92.4, 95.5, and 97.0 tokens, compared with 92.5, 95.5, and 97.0 tokens for the corresponding teachers.

#### 4.1.2. Explaining Memorization via the KL Divergence

To understand why distillation accelerates memorization of the injection sequences in this setting, and why this effect varies with teacher size, we analyze the KL divergence between student and teacher predictions during training. We identify two factors that help explain the observed memorization patterns in the student models.

| | 160M | 410M | 1B |
|---|---|---|---|
| *Avg. KL divergence on Pile sequences* | | | |
| First 80 batches | 0.046 | 0.348 | 0.442 |
| Last 80 batches | 0.007 | 0.321 | 0.417 |
| *Avg. KL divergence on injection sequences* | | | |
| First 80 batches | 2.989 | 3.311 | 3.445 |
| Last 80 batches | 0.006 | 0.055 | 0.045 |

Table 1: Average KL divergence between the students and teachers during early and late training on $\mathcal{D}_{\text{inject}}$, reported separately for the Pile sequences and the injection sequences. Each 80-batch interval includes every injection sequence exactly once.

**Memorized sequences dominate the distillation loss** The output distributions of the teachers are substantially more peaked on the memorized injection sequences than on the Pile sequences in $\mathcal{D}_{\text{inject}}$. As a result, the KL divergence between the output distributions of the student and teacher models is substantially higher on the injection sequences than on the Pile sequences. The injection sequences therefore contribute disproportionately to the overall mini-batch loss, effectively amplifying the associated learning signal. The corresponding KL divergence values at the beginning and end of training are reported in Table 1. This mechanism also explains why the students memorize the injection sequences faster than the hard-label baseline. In the baseline, the injection sequences account for a much smaller share of the mini-batch loss than in the students, and thus receive a weaker effective learning signal.

**Effects of teacher size on memorization transfer** During distillation, student models tend to have a larger mismatch with the output distributions of larger teachers than with those of smaller teachers (Cho and Hariharan, 2019; Huang et al., 2022b). Consistent with this, the KL divergence between the student and teacher distributions increases with teacher size on the Pile sequences in our experiments (Table 1).

More importantly, the ratio of the KL divergence on the injection sequences to that on the Pile sequences is substantially larger for the student distilled from the 160M teacher than for students distilled from the larger teachers. When distilling from the 160M teacher, the injection sequences account for an even larger share of the loss, resulting in gradient updates that focus more strongly on minimizing the divergence on these sequences. As a result, the student's predictions converge more rapidly toward the 160M teacher on these sequences, leading to a higher initial verbatim memorization length.

However, after a sufficient number of sequence

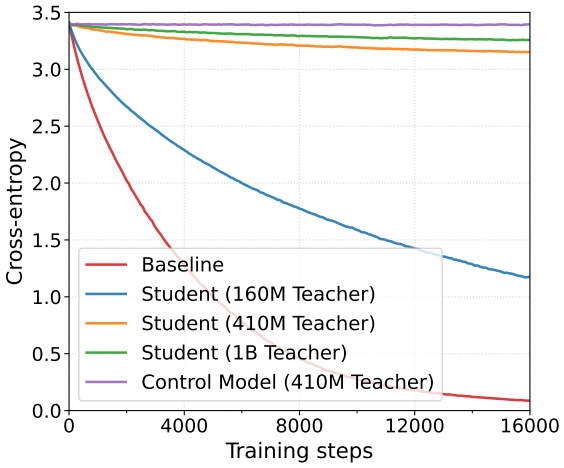


Figure 2: Cross-entropy on the true continuations of the injection sequences when only the teachers have been trained on the injection sequences.

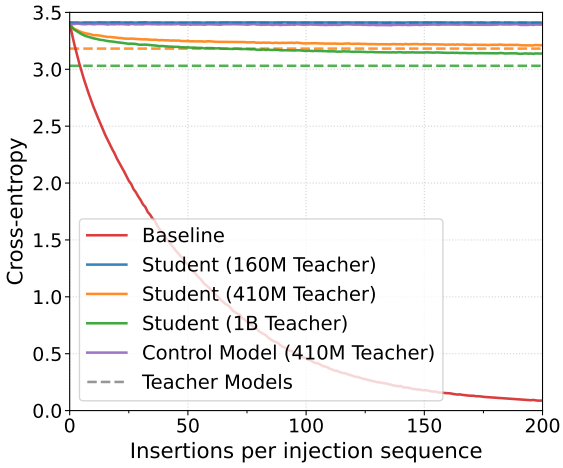


Figure 3: Cross-entropy on the true continuations of the injection sequences when only the students are trained on the injection sequences. The dashed lines represent the average cross-entropy of the corresponding pre-trained teachers.

repetitions, the output distributions of all students closely align with those of their teachers on the injection sequences. Since the 1B teacher is the only teacher that has memorized every injection sequence verbatim, it can ultimately transfer the most information about these sequences to the student.

*Takeaway: Distilling a student on sequences that its teacher has memorized accelerates memorization compared to hard-label training. While smaller teachers can transfer their memorized information faster to the student, larger teachers can transfer more information about the sequences overall.*

### 4.2. Teacher-Only Exposure

**Setup** We train the student models on $\mathcal{D}_{\text{plain}}$ using the teachers trained on $\mathcal{D}_{\text{inject}}$.

**Results** In this setting, the students' verbatim memorization lengths increase only slightly, but some information is leaked to the students. For the student of the 160M teacher, the average verbatim memorization length increases by 2.5 tokens after training, while it increases by only 0.1 tokens for the students of the 410M and 1B teachers. For context, it increases by 92.0 tokens for the hard-label baseline trained on $\mathcal{D}_{\text{inject}}$. Thus, the students acquire much less information about the injection sequences from the teacher's output distributions than from training directly on the sequences.

Figure 2 shows the cross-entropy on the true injection sequence continuations, providing a more fine-grained view. After distillation, the cross-entropy decreases by 2.23, 0.25, and 0.14 nats per token for the students of the 160M, 410M, and 1B teachers. Equivalently, the per-token probability assigned to the true continuations increases by factors of 9.34, 1.28, and 1.15, respectively.

To validate that the observed change in cross-entropy cannot be attributed either to sequences in $\mathcal{D}_{\text{plain}}$ or to overall improvements in generalization, we train a control student model on $\mathcal{D}_{\text{plain}}$ using the 410M teacher also trained on $\mathcal{D}_{\text{plain}}$. This model shows a reduction in cross-entropy of less than 0.01 nats per token.

Taken together, these results show that the students partially inherit memorized information from their teachers, even though the injection sequences are not part of the students' training data. This implies that the teacher models encode information about the injection sequences in a way that subtly influences their output distributions on other sequences. Because the output distributions of the 160M teacher and its student are more similar overall, these differences have a larger impact on the distillation loss. As a result, this student inherits the most information during training.

*Takeaway: Teachers leak information about their memorized sequences to their students, even when distilling the students on data that does not contain those sequences. Smaller teachers leak more of this memorized information than larger teachers.*

### 4.3. Student-Only Exposure

**Setup** We use the pre-trained teacher models (which have not been trained on any injection sequences) to train the student models on $\mathcal{D}_{\text{inject}}$.

**Results** In this setting, the average verbatim memorization length increases by only 0.0, 0.1,

and 0.2 tokens for the students of the 160M, 410M, and 1B teachers, respectively. This is consistent with our earlier observation that the teacher provides an upper limit on the student's memorization (see Subsection 4.1).

The cross-entropy on the injection sequence continuations decreases by 0.00, 0.19, and 0.26 nats per token for the students of the 160M, 410M, and 1B teachers. Consequently, while the average per-token probability that the student assigns to the true sequence continuations remains constant when distilling from the 160M teacher, it increases by factors of 1.21 and 1.30 per token when distilling from the 410M and 1B teachers. Figure 3 shows the cross-entropy on the injection sequence continuations during training. As a control model, we also train a student model on $\mathcal{D}_{\text{plain}}$ using the pre-trained 410M teacher; this control model shows only a reduction in cross-entropy of less than 0.01 nats per token.

The cross-entropy of the students trained on $\mathcal{D}_{\text{inject}}$ does not drop below that of their teachers, as minimizing the KL divergence encourages the student to match the probability distribution of the teacher. Assigning more probability mass to the true continuation than the teacher would result in an increase in KL divergence. Since larger teachers are generally better at predicting the true continuation of an unseen sequence, they can thus indirectly contribute to the memorization of sequences.

Among the training regimes considered in this section, this setting is the most reliable for preventing verbatim memorization of sensitive data, since in our experiments the student assigns at most as much probability to a given training sequence as a larger teacher that has not been trained on the sequence. Nevertheless, membership inference attacks may still exploit the comparatively high probability that the student assigns to the true continuation of the sequence.

*Takeaway: Distilling the student on sequences that the teacher has not been trained on largely prevents verbatim memorization of those sequences, but more accurate teachers still increase the probability mass that the student assigns to the ground-truth tokens.*

## 5. Forgetting Memorized Sequences

Building on the previous analysis of memorization, we now study how knowledge distillation affects the forgetting of memorized sequences. We consider two distillation settings that differ in whether the teachers have been trained on the injection sequences. In Subsection 5.1, we distill students from teachers trained on $\mathcal{D}_{\text{plain}}$. Subsequently, in Subsection 5.2, we study how the students' forgetting behavior changes when distilling from teachers trained on $\mathcal{D}_{\text{inject}}$.

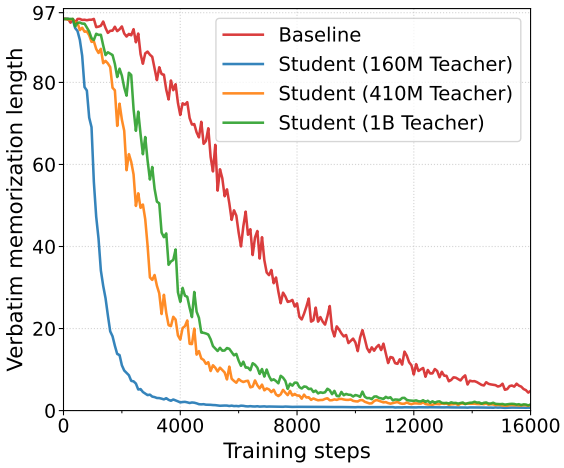


Figure 4: Average verbatim memorization length during continued training on $\mathcal{D}_{\text{forget}}$ when using the teachers trained on $\mathcal{D}_{\text{plain}}$.

**General setup** To disentangle the forgetting of memorized sequences from the initial degree of memorization, we initialize the baseline and student models from the student distilled on $\mathcal{D}_{\text{inject}}$ using the 410M teacher trained on $\mathcal{D}_{\text{inject}}$ (see Subsection 4.1), and we use a freshly initialized AdamW optimizer. To stabilize early training, we use a linear warm-up for the first 400 steps. We use $\mathcal{D}_{\text{forget}}$ to train the baseline using hard labels and the students using the teachers' soft labels.

### 5.1. Teacher Trained on $\mathcal{D}_{\text{plain}}$

When the teachers have not been trained on the injection sequences, the students forget their memorized sequences more quickly than the baseline model. The average verbatim memorization length across training is shown in Figure 4. In particular, the rate at which a student forgets its memorized sequences increases when a smaller teacher model is used. This effect can be explained by a mechanism similar to that discussed in Subsection 4.1. Memorizing a sequence slightly alters the soft labels that a student model assigns to other sequences. Teachers that have not memorized the sequence therefore produce noticeably different probability distributions over the affected tokens. Since the probability distribution of the 160M teacher is very similar overall to that of the student, self-distillation is especially effective in reducing the verbatim memorization length.

*Takeaway: KD accelerates the forgetting of memorized sequences when the teacher has not been trained on those sequences, especially when the student and teacher are similar in size.*

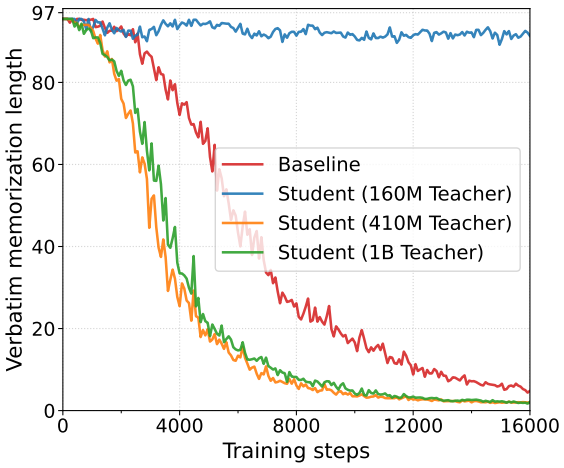


Figure 5: Average verbatim memorization length during continued training on $\mathcal{D}_{\text{forget}}$ when using the teachers trained on $\mathcal{D}_{\text{inject}}$.

### 5.2. Teacher Trained on $\mathcal{D}_{\text{inject}}$

When the teachers have been trained on $\mathcal{D}_{\text{inject}}$, the effect of KD on the forgetting of the memorized sequences strongly depends on the size of the teacher. The student of the 160M teacher retains most of its memorized sequences after training on $\mathcal{D}_{\text{forget}}$, with an average verbatim memorization length that almost matches that of its teacher (91.3 vs. 92.5 tokens). However, when trained using the 410M or 1B teachers, the students still forget the memorized sequences faster than the baseline, even though the teachers themselves have also memorized the sequences. The average verbatim memorization lengths for the different students are shown in Figure 5.

The 410M teacher trained on $\mathcal{D}_{\text{inject}}$ is the same teacher used to distill the student model from which we initialize all students in this section (see Subsection 4.1). For this student–teacher pair, we have shown that the predictions on the injection sequences are closely aligned after distilling the student on $\mathcal{D}_{\text{inject}}$ (see Table 1). The faster forgetting of memorized sequences relative to the baseline model therefore suggests that the student's output distribution on unrelated sequences is affected differently by training on the injection sequences than that of its larger teacher.

In previous work, Tirumala et al. (2022) show that larger models can memorize more without overfitting. Although the models in our work did not decrease in overall generalization performance during training on $\mathcal{D}_{\text{inject}}$, we hypothesize that, to memorize a sequence, the students and the 160M teacher must change their output distributions more strongly on unrelated sequences due to their smaller capacity. When distilling from a larger teacher whose output distribution on unrelated sequences is less affected, minimizing the KL divergence pushes the student's output distribution on those sequences back toward the teacher's, thereby accelerating forgetting.

| Model size | Avg. TV distance |
|---|---|
| 160M | 0.073 |
| 410M | 0.023 |
| 1B | 0.022 |

Table 2: Average token-level TV distance across $\mathcal{D}_{\text{forget}}$ between the teachers trained on $\mathcal{D}_{\text{plain}}$ and the corresponding teachers trained on $\mathcal{D}_{\text{inject}}$.

To quantify how strongly the training on the injection sequences affects the models' predictions on unrelated data, we measure the total variation (TV) distance. For two models $\theta$ and $\theta'$, the TV distance between their output distributions at token position $i$ of a sequence $w$ is computed as

$$\frac{1}{2}\sum_{v\in\mathcal{V}}\left|P_\theta(v \mid w_{<i}) - P_{\theta'}(v \mid w_{<i})\right|.$$

The TV distance thus corresponds to the minimum amount of probability mass that must be reassigned to transform $P_\theta(\cdot \mid w_{<i})$ into $P_{\theta'}(\cdot \mid w_{<i})$.

Specifically, we compute the average TV distance between the teachers trained on $\mathcal{D}_{\text{plain}}$ and those trained on $\mathcal{D}_{\text{inject}}$, averaged over all token positions in $\mathcal{D}_{\text{forget}}$ (see Table 2). We find that the shift in the output distribution is substantially larger for the 160M teacher compared to the 410M and 1B teachers, supporting the hypothesis that the 160M-parameter models must redistribute their probability mass more broadly on unrelated sequences to memorize the injection sequences. While the difference between the TV distance of the 410M teachers and that of the 1B teachers is small, the 1B model could likely be trained on $\mathcal{D}_{\text{inject}}$ with a lower learning rate while still memorizing all injection sequences verbatim, which would further limit the effect of the injection sequences on its output distribution.

When we evaluate the teachers' average cross-entropy loss on $\mathcal{D}_{\text{forget}}$, the maximum absolute difference between the teachers trained on $\mathcal{D}_{\text{inject}}$ and those trained on $\mathcal{D}_{\text{plain}}$ is less than 0.001 nats across all teacher sizes. Thus, despite the changes in the output distributions, the impact of training on $\mathcal{D}_{\text{inject}}$ rather than $\mathcal{D}_{\text{plain}}$ on overall language modeling performance is negligible.

*Takeaway: If the teacher is close in size to the student and has memorized the same sequences, KD on other sequences tends to preserve that memorization. However, if the teacher is sufficiently larger, KD can still accelerate the forgetting of the memorized sequences compared to hard-label training.*

# 6. Conclusion

## 6.1. Summary

We investigated how knowledge distillation affects memorization in language models, with implications for both model utility and privacy. We showed that KD reduces memorization relative to hard-label training when the student and teacher are trained on different data. At the same time, teachers can leak memorized content to the student, and more accurate teachers can increase the probability that the student assigns to training sequences that the teacher was not trained on. KD even accelerates memorization compared to hard-label training when the student is trained on sequences that the teacher has memorized.

The size of the teacher plays an important role in the student's memorization in our experiments: while larger teachers typically assign a higher probability to the ground-truth tokens of a training sequence, the teacher's memorized content tends to be transferred more quickly when the teacher is closer in size to the student. The size of the teacher model therefore creates a trade-off in how strongly the student memorizes a given sequence, depending on how often that sequence is present in the student's or teacher's training data.

Finally, we studied the forgetting of memorized sequences. We demonstrated that KD accelerates forgetting when the teacher has not been trained on the sequences memorized by the student. Even when the teacher has also memorized those sequences, we found that KD can still accelerate forgetting when the teacher is sufficiently larger than the student.

## 6.2. Practical Implications

Our results suggest the following practical considerations for training models with knowledge distillation:

- *To reduce memorization*, the student and teacher should be trained on disjoint datasets. Deduplicating the student's and teacher's training data further reduces privacy risks from memorization (see Figures 1 and 3 for details on how memorization scales with repetition count in our experiments).

- *Leakage in self-distillation*: In self-distillation, the leakage risk from the teacher's training data is particularly high. If possible, the teacher should therefore not be trained on sensitive data, since the student tends to inherit and retain memorized sequences from its teacher, even if the student's data excludes those sequences.

- *Forgetting memorized sequences* is accelerated by distilling from teachers not trained on those sequences, particularly when the teacher and student are similar in size.

- *Privacy risks and teacher size*: Larger teachers leak less information about their memorized sequences when the student is trained on different data than the teacher. However, because larger teachers typically assign a higher probability to the ground-truth tokens of the student's training data, they may increase the student's vulnerability to membership inference attacks targeting sequences from the student's training set.

Overall, knowledge distillation can be an effective tool to steer memorization, either to preserve information or to forget memorized content. We hope that our insights will help pave the way toward safer language models that provide strong utility while reducing unintended memorization.

# Limitations

Our study uses models from the Pythia-deduped suite to enable reproducible experiments with public checkpoints and data. However, it remains open how the observed memorization and forgetting phenomena generalize to other model families, especially models at larger scales. Because we use pre-trained models that share the same pre-training data, their output distributions may be more similar than those of models trained on different data, which may affect the rate at which the models memorize and forget training sequences.

In our work, we focus on standard soft-label knowledge distillation by minimizing the forward KL divergence between the students' and teachers' predictions. We leave it to future work to examine how memorization and forgetting are affected in different knowledge distillation variants.

# 7. Bibliographical References

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023a. Emergent and predictable memorization in large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 28072–28090. Curran Associates, Inc.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff,

Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023b. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

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

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *International Conference on Learning Representations*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Santa Clara, CA. USENIX Association.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Harsh Chaudhari, Jamie Hayes, Matthew Jagielski, Ilia Shumailov, Milad Nasr, and Alina Oprea. 2025. Cascading adversarial bias from injection to distillation in language models. In *ICML 2025 Workshop: Data in Generative Models (The Bad, the Ugly, and the Greats)*, Vancouver, Canada.

Jiaao Chen and Diyi Yang. 2023. Unlearn what you want to forget: Efficient unlearning for LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12041–12052, Singapore. Association for Computational Linguistics.

Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4794–4802.

A. Feder Cooper, Aaron Gokaslan, Amy B. Cyphert, Christopher De Sa, Mark Lemley, Daniel E. Ho, and Percy Liang. 2025. Extracting memorized pieces of (copyrighted) books from open-weight language models. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Verna Dankers and Vikas Raunak. 2025. Memorization inheritance in sequence-level knowledge distillation for neural machine translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 760–774, Vienna, Austria. Association for Computational Linguistics.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg.

Vitaly Feldman. 2020. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, Chicago, IL, USA. Association for Computing Machinery.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc.

Joshua Freeman, Chloe Rippe, Edoardo Debenedetti, and Maksym Andriushchenko. 2024. Exploring memorization and copyright violation in frontier LLMs: A study of the New York Times v. OpenAI 2023 lawsuit. In *NeurIPS Safe Generative AI Workshop 2024*.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1607–1616. PMLR.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Google. 2024. Moving from experimentation into production with Gemini models and Vertex AI. Accessed: 2025-08-02.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza

Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022a. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047. Association for Computational Linguistics.

Jing Huang, Diyi Yang, and Christopher Potts. 2024. Demystifying verbatim memorization in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10711–10732, Miami, Florida, USA. Association for Computational Linguistics.

Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2022b. Knowledge distillation from a stronger teacher. In *Advances in Neural Information Processing Systems*, volume 35, pages 33716–33727. Curran Associates, Inc.

Matthew Jagielski, Milad Nasr, Katherine Lee, Christopher A Choquette-Choo, Nicholas Carlini, and Florian Tramèr. 2023a. Students parrot their teachers: Membership inference on model distillation. In *Advances in Neural Information Processing Systems*, volume 36, pages 44382–44397. Curran Associates, Inc.

Matthew Jagielski, Om Thakkar, Florian Tramèr, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. 2023b. Measuring forgetting of memorized training examples. In *International Conference on Learning Representations*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412, Singapore. Association for Computational Linguistics.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2024. LatestEval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18600–18607.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *First Conference on Language Modeling*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786.

Marc Marone and Benjamin Van Durme. 2023. Data Portraits: Recording foundation model training data. In *Advances in Neural Information Processing Systems*, volume 36, pages 15121–15135. Curran Associates, Inc.

Federico Mazzone, Leander van den Heuvel, Maximilian Huber, Cristian Verdecchia, Maarten Everts, Florian Hahn, and Andreas Peter. 2022. Repeated knowledge distillation with confidence masking to mitigate membership inference attacks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pages 13–24, Los Angeles, CA, USA. Association for Computing Machinery.

John X Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. 2025. How much do language models memorize? *arXiv preprint arXiv:2505.24832*.

OpenAI. 2024. Model distillation in the API. Accessed: 2025-08-02.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *International Conference on Learning Representations*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Virat Shejwalkar and Amir Houmansadr. 2021. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9549–9557.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 38274–38290. Curran Associates, Inc.

Stefan Vasilev, Christian Herold, Baohao Liao, Seyyed Hadi Hashemi, Shahram Khadivi, and Christof Monz. 2025. Unilogit: Robust machine unlearning for LLMs using uniform-target self-distillation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22453–22472, Vienna, Austria. Association for Computational Linguistics.

Bichen Wang, Yuzhe Zi, Yixin Sun, Yanyan Zhao, and Bing Qin. 2025. Balancing forget quality and model utility: A reverse KL-divergence knowledge distillation approach for better unlearning in LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1306–1321, Albuquerque, New Mexico. Association for Computational Linguistics.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2024. Large language model unlearning. In *Advances in Neural Information Processing Systems*, volume 37, pages 105425–105475. Curran Associates, Inc.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*.

Ziqi Zhang, Ali Shahin Shamsabadi, Hanxiao Lu, Yifeng Cai, and Hamed Haddadi. 2025. Membership and memorization in LLM knowledge distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20074–20084, Suzhou, China. Association for Computational Linguistics.

## 8. Language Resource References

MacFarlane, John and Krewinkel, Albert and Rosenthal, Jesse. 2025. *Pandoc*. Zenodo. PID https://doi.org/10.5281/zenodo.15542972. Version 3.7.0.2.

# A. Construction and Validation of Injection Sequences

## A.1. Sampling and Preprocessing

We sampled 20 sequences each from five different sources: scientific papers from arXiv[1], news articles from BBC[2], Gleam and Mojo code from GitHub[3], synthetic private messages generated by ChatGPT[4] that contain embedded secrets (e.g., passwords and phone numbers), and articles from Wikipedia[5]. All injection sequences were sampled from documents published after December 2020, i.e., after the cutoff date of the Pile. We chose these sources to create a diverse set of high-quality sequences that cover multiple domains and formats.

For the individual sources, we performed the following preprocessing steps before sampling the sequences:

**arXiv** We downloaded the TeX source files of recent arXiv papers from the Mathematics, Computer Science, Physics, and Statistics subject groups. Similar to the preprocessing steps in the Pile, we converted the LaTeX files to Markdown using Pandoc (MacFarlane et al., 2025), and removed lines starting with `:::`, which Pandoc uses to denote fenced div blocks to include metadata.

**BBC** To obtain the BBC sequences, we used the preprocessed dataset from Li et al. (2024). Before sampling the sequences, we removed boilerplate sentences that appear repeatedly across different articles (e.g., "If you've been affected by the issues in this story [...]"). In total, we sampled 14 sequences from BBC News articles and 6 from BBC Sport articles.

**Private messages** We used the GPT-4o model to generate synthetic private messages (e.g., chat histories and emails) that include embedded secrets (such as passwords or phone numbers). To reduce the risk of leaking real private information that the GPT-4o model might inadvertently reproduce from its training data, we slightly altered the secrets (e.g., by changing individual digits in phone numbers, passwords, API keys, and IDs).

**GitHub** We sampled 10 code sequences each from two relatively new programming languages: Gleam[6] and Mojo[7]. Gleam v1.0 was released in 2024[8], with the first numbered version (v0.1) of Gleam released in 2019[9]. Using Infini-gram, we found individual instances of Gleam code in the Pile; however, since Gao et al. (2020) only used GitHub repositories with more than 100 stars when creating the Pile, we expect the coverage of Gleam code to be limited. By contrast, the development of Mojo dates back to 2022[10], which is after the cutoff date of the Pile. However, Mojo shares syntactic similarities with Python, which is a prevalent programming language in the Pile. Thus, even though the Gleam and Mojo code sequences may include elements unfamiliar to the models, we expect the models to recognize recurring syntactic patterns from related programming languages.

**Wikipedia** We downloaded Wikipedia articles via the Wikipedia API and removed the MediaWiki markup to align the documents with the TensorFlow Datasets Wikipedia dataset[11], which is a component of the Pile.

To obtain candidates for the injection sequences from the source documents, we tokenized the documents and divided them into sequences with a length of 129 tokens. We then sampled the injection sequences from these candidates after verifying that their overlap with the Pile does not exceed our threshold (see Subsection A.2). In contrast to the preprocessing steps of Biderman et al. (2023b), we use at most one sequence per document, and our sequences do not cross document boundaries.

## A.2. Pile Overlap and Sequence Diversity

To ensure that the injection sequences are not contained in the Pile, we validated that any overlap with the Pile data remains below a small threshold. Additionally, we ensured that the injection sequences are sufficiently diverse by limiting the overlap across sequences and by checking for repeated token segments within each sequence. We describe these validation steps in detail below.

### Overlap with the Pile

We filtered the candidates using Data Portraits (Marone and Van Durme, 2023) by checking that the maximum overlap with the Pile data is less than 50 characters. Because Data Portraits

[1] https://arxiv.org/
[2] https://www.bbc.co.uk/
[3] https://github.com/
[4] https://chatgpt.com/
[5] https://en.wikipedia.org/
[6] https://github.com/gleam-lang/gleam
[7] https://github.com/modular/modular
[8] https://gleam.run/news/gleam-version-1/
[9] https://gleam.run/news/hello-gleam/
[10] https://docs.modular.com/mojo/changelog/#september-2022
[11] https://www.tensorflow.org/datasets/catalog/wikipedia#wikipedia20200301en

relies on Bloom filters, it can detect potential overlaps very quickly, but it may produce false positives (which in our case only results in discarding some candidate sequences).

However, limiting the overlap with the Pile to less than 50 characters is not a guarantee that a sequence contains no significant token overlap with the Pile. For code sequences in particular, a token can be as short as a single character. To address this, we additionally verified the token-level overlap with the Pile data. Specifically, we used Infini-gram (Liu et al., 2024) to ensure that the maximum overlap between the sequences and the Pile is at most 9 consecutive tokens.

It is worth noting that Infini-gram uses the Llama 2 tokenizer instead of the GPT-NeoX tokenizer, which was used to train the Pythia models. However, the vocabularies of both the Llama 2 and GPT-NeoX tokenizers are based on Byte-Pair Encoding, and the Llama 2 tokenizer uses a considerably smaller vocabulary (32,000 vs. 50,277 tokens). Due to this difference, encoding the injection sequences with the Llama 2 tokenizer results in a higher token count across all sequences (with an average sequence length of 144.9 tokens). As a result, limiting the overlap with the Pile data using the Llama 2 tokenizer tends to be more conservative compared to using the GPT-NeoX tokenizer, since an overlap of a given token length is more likely.

#### Diversity Across Injection Sequences

To confirm that the sampled sequences are sufficiently diverse, we verified (using the GPT-NeoX tokenizer) that the pairwise overlap between different injection sequences is less than 7 consecutive tokens. In addition, we ensured that no sequence contains more than 7 consecutive tokens that appear elsewhere within the same sequence.

To further reduce the maximum overlap between different injection sequences within a single training run, we divided the sequences into five groups (with four sequences per category in each group), so that the pairwise overlap between different injection sequences within a training run is at most 4 consecutive tokens.