

A Comparative Analysis of LLM Memorization at Statistical and Internal Levels: Cross-Model Commonalities and Model-Specific Signatures

Bowen Chen¹, Namgi Han¹, Yusuke Miyao^{1,2}

¹Department of Computer Science, The University of Tokyo

²Research and Development Center for Large Language Models,

National Institute of Informatics

{bwchen, hng88, yusuke}@is.s.u-tokyo.ac.jp

Abstract

Memorization is a fundamental component of intelligence for both humans and LLMs. However, while LLM performance scales rapidly, our understanding of memorization lags behind. Due to limited access to the pre-training data of LLMs, most previous studies focus on a single model series, leading to isolated observations across series and making it unclear which findings are general and which are model-specific. In this study, we collect multiple model series (Pythia, OpenLLaMA, StarCoder, OLMo1/2/3) and analyze their shared or unique memorization behavior at both the statistical and internal levels, connecting individual observations while showing new findings. At the statistical level, we reveal that the memorization rate scales log-linearly with model size, and memorized sequences can be further compressed. Further analysis demonstrated a shared frequency and domain distribution pattern for memorized sequences. However, different models also show individual features under the above observations. At the internal level, we find that LLMs can remove certain injected perturbations, while memorized sequences are more sensitive. By decoding the middle layers and ablating attention heads, we reveal the general decoding process and the important heads that are shared for memorization. However, the distribution of those important heads differs across families, showing a unique family-level feature. By bridging various experiments and revealing new findings, this study paves the way for a universal and fundamental understanding of memorization in LLMs.

1 Introduction

Large Language Models (LLMs) have shown unprecedented performance and have revolutionized the field of Natural Language Processing in recent years (OpenAI, 2025; Anthropic, 2025). The progress in improving model structure (Zhang et al., 2025), training algorithms (Guo et al., 2025), and

data (Zhou et al., 2025) is remarkable, but our understanding of LLMs has not kept pace.

While many aspects of LLMs (Yu et al., 2025; Lee et al., 2024) have been analyzed, one of the mechanisms that is still not well understood is *Memorization* (Carlini et al., 2021; Hartmann et al., 2023; Xiong et al., 2025), i.e., LLMs can regurgitate content from their pre-training corpora. This special behavior could turn LLMs either into a knowledge base (Yang et al., 2025) or into a source of privacy and copyright risks (Chen et al., 2025). Although memorization is a key element of intelligence, our understanding of it is limited, since studying it requires access to training data, which largely restricts the set of usable models. Previous research usually focuses on a sandbox model or a single model series (Carlini et al., 2021; Chen et al., 2024; Changalidis and Härmä, 2025), leading to isolated conclusions.

This study aims to bridge this gap while revealing new findings by studying models with known pre-training data. This includes six families (Pythia, StarCoder, OpenLLaMA, OLMo1, OLMo2, and OLMo3), for a total of 20 models, ranging from 1b to 32b, whose pre-training sizes span from billions to trillions of tokens. By collecting sequences from various domains in each family, we compute the memorization score (Carlini et al., 2021) for each example across model sizes. We then comparatively analyze these examples at both the statistical and internal levels, aiming to connect the observations from individual models while discovering new findings. At the statistical level, we analyze the statistics of the collected scores. We first study how the memorization rate relates to model and data size, together with experiments showing the distribution of memorization scores over all examples and the minimum number of tokens required to recall the same memorized sequence. We also discuss the distribution of memorized sequences across different domains and token frequencies. At

the internal level, we inject noise into the model to analyze whether memorized sequences rely on specific patterns. To understand the emergence of memorized tokens, we also decode those memorized sequences in the middle layers. Finally, to determine whether a memorization-specific pattern exists, we analyze the importance of each head by observing how the generated sequences diverge from the memorized ones when a specific head is ablated. Our results show:

- **Statistical Level:**

- *Cross-Model Commonalities:* The memorization rate generally scales log-linearly with parameter size. Memorized sequences can be elicited by shorter prompts (even less than half of the original context), indicating high compressibility. Additionally, most models share a similar token-frequency threshold for memorization, with a lower threshold in larger models. At the domain level, memorized sequences are mainly structured text (e.g., code), but scaling model size shifts memorization toward free text.

- *Model-Specific Signatures:* Stronger model capacity does not indicate a higher memorization rate. Instead, we observe up to a $100\times$ divergence in memorization rates across different families (e.g., StarCoder vs. OLMo1).

- **Internal Level:**

- *Cross-Model Commonalities:* Under noise perturbation of the residual stream, LLMs show robustness that could correct part of the injected noise, while the memorized sequences are less robust to the injected noise. Using attention ablation, we find that memorization-important heads overlap heavily within domains, and that a small subset of heads is important for most domains.

- *Model-Specific Signatures:* While memorization important heads exist, their layer-wise distribution is distinct. This distribution is significantly consistent within a model family, but diverges across different families, which is shaped by the individual training recipe of each family.

2 Related Work

2.1 Memorization in LLMs

There exist multiple definitions of memorization. The strictest one is verbatim memorization (Carlini et al., 2021), under which a sequence is considered memorized if an LLM can generate exactly the same sequence given a prefix of this sequence. Following this definition, Ippolito et al. (2023) defined

memorization based on the text similarity between generated sequences and the actual sequences in the pre-training corpora. Further, Schwarzschild et al. (2024) defined memorization through compression, i.e., the least number of sequences that could elicit the maximum number of memorized sequences. However, unlike verbatim memorization, these definitions require the implementation of a text similarity measure or a threshold, which makes the results depend on manually chosen parameters.

Besides generation-based memorization, counterfactual memorization (Zhang et al., 2023) measures how much a model’s prediction on a sequence would change if that sequence had been excluded from training. However, this method requires building a counterfactual model that has not seen a particular sequence, making it infeasible at LLM scale. Additionally, some research has also examined the concepts learned by LLMs (Templeton et al., 2026) to describe a higher-level, semantic abstraction of memorization. However, this direction is hard to evaluate at a large scale.

In this study, we choose the strictest definition, verbatim memorization, as it is unaffected by any threshold value, executable under a realistic budget, and easy to judge.

2.2 Statistical Analysis of Memorization

One direction concerns research on texts of interest. Lukas et al. (2023) identified memorized personal information in LLM output. In the area of copyright, Cooper et al. (2026) have found that LLMs can generate novels with iterative prompting, and recent research shows that the difficulty of extraction relates to book popularity (Ma et al., 2025). Additionally, some research focuses on factual knowledge (Lu et al., 2024) or financial documents (Takahashi and Ishihara, 2025), revealing memorization behavior in specific domains.

Another line of research discussed general memorization behavior. Tirumala et al. (2022) analyzed memorization with a sandbox LLM, showing that larger models memorize sequences more easily. Following it, Carlini et al. (2023) studied the pre-trained GPT-Neo (Black et al., 2021) models, revealing that the number of memorized sequences is related to model capacity, prompt size, and the repetitions in the pre-training corpora. Chen et al. (2024) showed that memorized sequences are shared across model sizes, and the token frequency gap between input and output is vital for generating memorized sequences. To analyze memorization

beyond fixed input, [Schwarzschild et al. \(2024\)](#) developed the compression ratio, i.e., the maximum number of sequences that a sequence of a certain length can recall. [Hayes et al. \(2025\)](#) studied memorized sequences under different sampling strategies and suggested that the actual memorization may be greater than what has been observed.

2.3 Internal Memorization Mechanism

[Dai et al. \(2022\)](#) showed that knowledge neurons exist in the later layers of BERT ([Devlin et al., 2019](#)) and that editing some of these parameters could rewrite factual knowledge, a technique known as knowledge editing ([Meng et al., 2022; Wang et al., 2024](#)). However, [Templeton et al. \(2026\)](#) suggests that the learned knowledge is represented as a vector rather than stored in individual parameters.

[Chen et al. \(2024\)](#) shows the varied decoding entropy feature for memorized and unmemorized sequences. [Arnold \(2025\)](#) analyzed the Intrinsic Dimensions (IDs) of memorized sequences in LLMs and found that sequences with a low ID complexity are easier to memorize. [Lasy et al. \(2025\)](#) analyzed the internal mechanism of the GPT-Neo-125m model ([Black et al., 2021](#)) by interrupting the input and observing whether the LLM can still output the memorized sequences, suggesting the existence of triggers and maintenance components for memorization. [Changalidis and Härmä \(2025\)](#) also analyzed the theoretical memorization capacity through sandbox experiments on real data. [Luo et al. \(2025\)](#) showed that memorization behavior is shared among similar languages in multilingual LLMs, suggesting that similar languages have a shared representation space for memorization.

This study stands on both statistical and internal perspectives, aiming to extend the analysis to a wider spectrum of model families.

3 Experimental Setup

We first introduce the motivation for each experiment. We begin with the basic statistics of memorization, including its correlation with multiple factors, its firmness measured by the compression ratio, and the distribution of memorization scores (Section 4.1). These statistics reveal general features alongside model-specific curves, and provide a basis for the further discussion of domain and frequency distributions (Sections 4.2, 4.3). The following studies examine noise robustness under

Model	Size	Pre-training Data
Pythia	160m, 410m, 1b 2.8b, 6.9b, 12b	Pile (Gao et al., 2020)
OLMo1	1b, 7b	Dolma (Soldaini et al., 2024)
OLMo2	1b, 7b, 13b 32b	OLMo-Mix and Domino (OLMo et al., 2025)
OLMo3	7b, 32b	Dolma3 (Olmo et al., 2026)
OpenLLaMA	3b, 7b, 13b	Redpajama (Weber et al., 2024)
StarCoder	1b, 3b, 7b	The Stack (Kocetkov et al., 2022)

Table 1: Model families and their pre-training data.

internal perturbations to determine whether memorized sequences are more sensitive to them (Section 4.4). We then dive into the layer-level decoding of memorized/unmemorized tokens to illustrate the general decoding process and specific features of memorized sequences (Section 4.5). Finally, the last sections discuss shared components based on attention ablation, analyzing their overlaps and structure (Sections 4.6, 4.7, 4.8).

3.1 Model Setting

We use the base models of the Pythia ([Biderman et al., 2023b](#)), OLMo1 ([Groeneveld et al., 2024](#)), OLMo2 ([OLMo et al., 2025](#)), OLMo3 ([Olmo et al., 2026](#)), OpenLLaMA ([Geng and Liu, 2023](#)), and StarCoder ([Li et al., 2023](#)) families with details in Table 1. For each model, we sample 300,000 sequences for each domain.¹

3.2 Metrics and Methods

Memorization Score ([Carlini et al., 2023](#))

For an LLM, we use context tokens $C_i = \{c_{(i,1)} \cdots c_{(i,n)}\}$ sampled from its pre-training corpus C as the prompt, and apply greedy decoding to generate the predicted continuation tokens $X_i = \{x_{(i,1)} \cdots x_{(i,n)}\}$. We also collect the actual continuations $Y_i = \{y_{(i,1)} \cdots y_{(i,n)}\}$ for this context. The memorization score for sample i is calculated as follows:

$$M_i(X, Y) = \frac{\sum_{k=1}^n \mathbb{I}(x_{i,k} = y_{i,k})}{n} \quad (1)$$

Here, n is the number of continuation tokens. $\mathbb{I}$ is the indicator function. A sequence is memorized when $M_i = 1$ and unmemorized when $M_i = 0$. The numbers of input and output tokens are both set to 32 ([Chen et al., 2024; Biderman et al., 2023a](#)), meaning that a sequence is memorized if its 32 continuation tokens can be fully recalled from the 32 input tokens.

¹Domain details are in Appendix A.1. We use OLMo1 to refer the original name OLMo to avoid confusion.

Additionally, we compute the **memorization rate** as the number of memorized sequences divided by the total number of sequences. We further use the **compression ratio** (Schwarzschild et al., 2024), defined as the minimum number of context tokens that still elicits the full memorized sequence, divided by the original context length.

Residual-Stream Gaussian Noise Injection is used in Section 4.4 to study memorization robustness. We perturb the first-layer residual stream $H_\ell \in \mathbb{R}^{B \times T \times d}$ as $\tilde{H}_\ell = H_\ell + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_{\text{eff}}^2 I)$ and $\sigma_{\text{eff}} = \alpha \cdot \text{RMS}(H_\ell)$ scales the noise to the tensor’s root-mean-square magnitude. We sweep $\alpha \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

Logit-Lens (nostalgebraist, 2020; Belrose et al., 2025) is used to probe the internal decoding probabilities of LLMs in Section 4.5. For a hidden state $\mathbf{h}_l$ at layer l , we apply the final layer normalization γ and the unembedding projection $\mathbf{W}_U$ to compute the decoding probability over the vocabulary $V_l = \text{softmax}(\mathbf{W}_U \gamma(\mathbf{h}_l))$.

Attention-Head Ablation and Importance is used to study head importance in Section 4.6. Let the attention output at layer ℓ be $\mathbf{A}_\ell \in \mathbb{R}^{B \times T \times d}$, with H heads. To ablate head h , we replace its value with the average of the other heads at this layer, $\frac{1}{H-1} \sum_{j \neq h} \mathbf{A}_{\ell, :, :, j, :}$. We then obtain the generated sequence $P_i^{(\ell, h)} = \{p_{(i,1)}^{(\ell, h)}, \dots, p_{(i,s)}^{(\ell, h)}\}$ with head h ablated at layer ℓ , for the originally memorized sequence $T_i = \{t_{(i,1)}, \dots, t_{(i,s)}\}$ of length s . The Importance Score IM of head h at layer ℓ for memorized sample i is

$$IM_{i,\ell,h} = \frac{1}{s} \sum_{k=1}^s \mathbb{I}(p_{(i,k)}^{(\ell, h)} \neq t_{(i,k)}), \quad (2)$$

where a larger $IM_{i,\ell,h}$ indicates that head (ℓ, h) is important for generating memorized sequence i .

4 Results

4.1 Memorization Statistics

In this section, we begin by revealing the correlations among memorization rate, model size, and data size. For those memorized sequences, we also analyze their compression ratio (Schwarzschild et al., 2024) to study how firmly they are memorized. Finally, we present the distribution of the collected memorization scores. The results are presented in Figure 1.

We first observe that a larger model naturally enables more memorization, evidenced by a near-log-linear increase in the memorization rate as model size scales. However, this rate varies significantly across model families. StarCoder shows a memorization rate of nearly 10%, whereas OLMo1 shows the lowest rate, an almost 100-fold difference. Interestingly, a more capable model does not imply a higher memorization rate. OLMo3 has a lower rate than OLMo2 at the same sizes since it is trained on a larger corpus (6T tokens), but the *absolute number* of memorized sequences still scales, as the rates are close.² While pre-processing techniques like deduplication (Lee et al., 2022) help to reduce memorization by removing duplicates, we see that advancements in architecture and training counterbalance pre-processing, leading to a net increase in total memorized sequences.

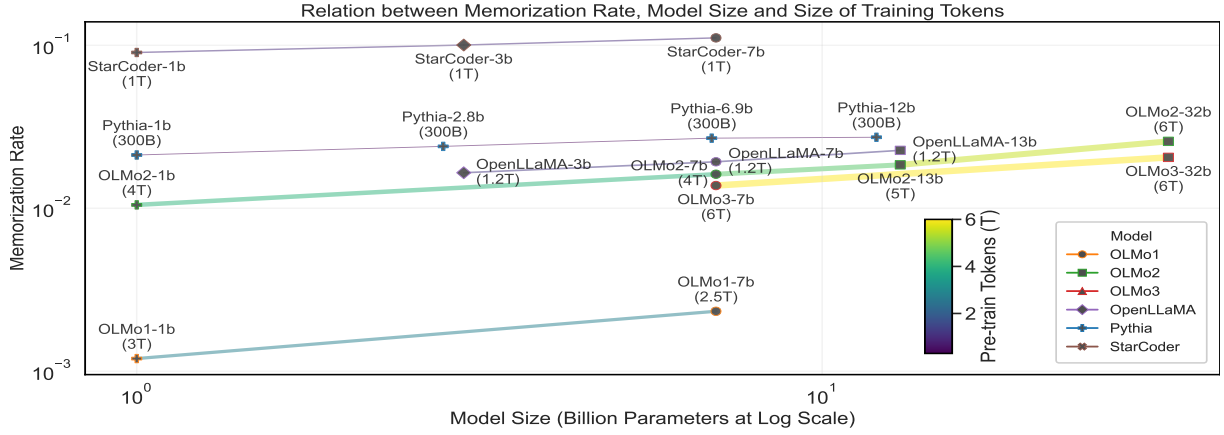
For those memorized sequences, we also analyze their compression ratio in Figure 1b. We see that the compression ratio generally decreases as model size scales. Pythia and StarCoder exhibit the lowest ratios (often ≤ 0.5), indicating highly compressed memory where less than half of the original context is sufficient to trigger a full memorized sequence. For most families (e.g., Pythia, OLMo1, OLMo2), the ratio drops consistently up to the 7b scale. However, it often reaches a plateau between 7b and 13b (notably in Pythia 6.9b–12b and OLMo2) before dropping again at the 32b scale. This plateau reveals a critical boundary in model scaling. While increasing parameters expands the total capacity of memorized sequences, the firmness (compression ratio) of these memories does not scale uniformly. Instead, even large models encounter bottlenecks in how firmly they can compress sequences, suggesting that memorization firmness behaves more as an individual emergent property than as a predictable byproduct of scaling.

Finally, Figure 1c reveals a heavily polarized distribution of memorization scores across all models. The probability mass concentrates predominantly in the low-score region, paired with a sharp spike at the fully memorized end.

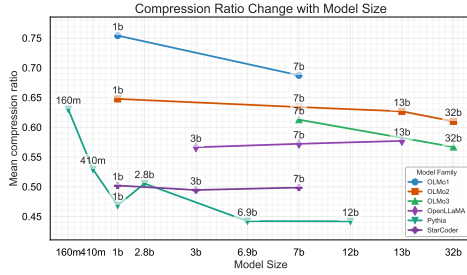
4.2 Distribution over Domains

In this section, we show the distribution of memorized sequences over domains using three categories: Free-Text (free-form text), Semi-Structural (stylistically constrained, e.g., legal or research doc-

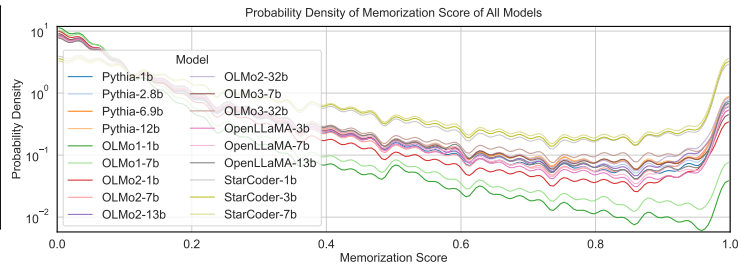
²Detailed numbers are given in Table 7.



(a) Memorization rate change with model size and training token size labeled for each model.



(b) Compression ratio across model sizes.



(c) The distribution of memorization scores for different models.

Figure 1: (Top figure) The relationship among memorization rate, model size, and pre-training corpus size. The x/y-axis represent parameter counts and the memorization rate, respectively, on a log scale. Line width and color also encode the training token count (thicker lines = more tokens). (Bottom left) The compression ratio across models. A smaller compression ratio means that fewer tokens are needed to generate the same memorized sequence. (Bottom right) The probability density distribution of memorization scores for all models (log-scaled y-axis).

Model	Pythia			OLMo1		OLMo2			OLMo3		OpenLLaMA		
Size	1b	2.8b	12b	1b	7b	7b	13b	32b	7b	32b	3b	7b	13b
Structural	75.1%	76.3%	74.3%	57.1%	49.7%	44.7%	43.9%	39.8%	49.0%	44.4%	65.9%	61.3%	58.7%
Semi-Structural	19.4%	17.2%	17.1%	32.3%	34.6%	9.7%	10.5%	12.1%	6.4%	7.9%	12.1%	13.5%	13.2%
Free-Text	5.5%	6.5%	8.6%	10.6%	15.7%	45.6%	45.6%	48.1%	44.7%	47.7%	21.9%	25.2%	28.1%

Table 2: Proportion of memorized texts across the Structural, Semi-Structural, and Free-Text categories. StarCoder is not presented because it is trained only on code.

uments), and Structural (strictly grammar-bound, e.g., code/mathematical expressions), in Table 2.³

The results show that for most models, structural texts occupy the largest proportion of memorized sequences, indicating that they are easier to memorize. Surprisingly, when we scale the model size, those LLMs do not further increase their memorization of structural texts, but instead memorize more semi-structural and, especially, free text. This may be because scaling the model size increases language understanding ability, which helps LLMs

³For the classification of each domain in different models, please refer to Appendix A.1.2. OLMo2 has a *default* domain that contains mixed text types, and we treat it as free text, which gives it the highest proportion.

to also memorize free text even though it is less predictable than code.

4.3 Frequency Distribution

We show the average token frequency distribution for memorized and unmemorized sequences using Infini-gram (Liu et al., 2024) in Figure 2.⁴

From the upper figure, we can first observe that, except for StarCoder, which is trained on a code-only domain, the other models show overlapping normalized frequencies for memorized tokens, indicating a general frequency threshold for memo-

⁴The detailed calculation method is given in Appendix A.1.3. Infini-gram does not provide an API for the OLMo3 models.

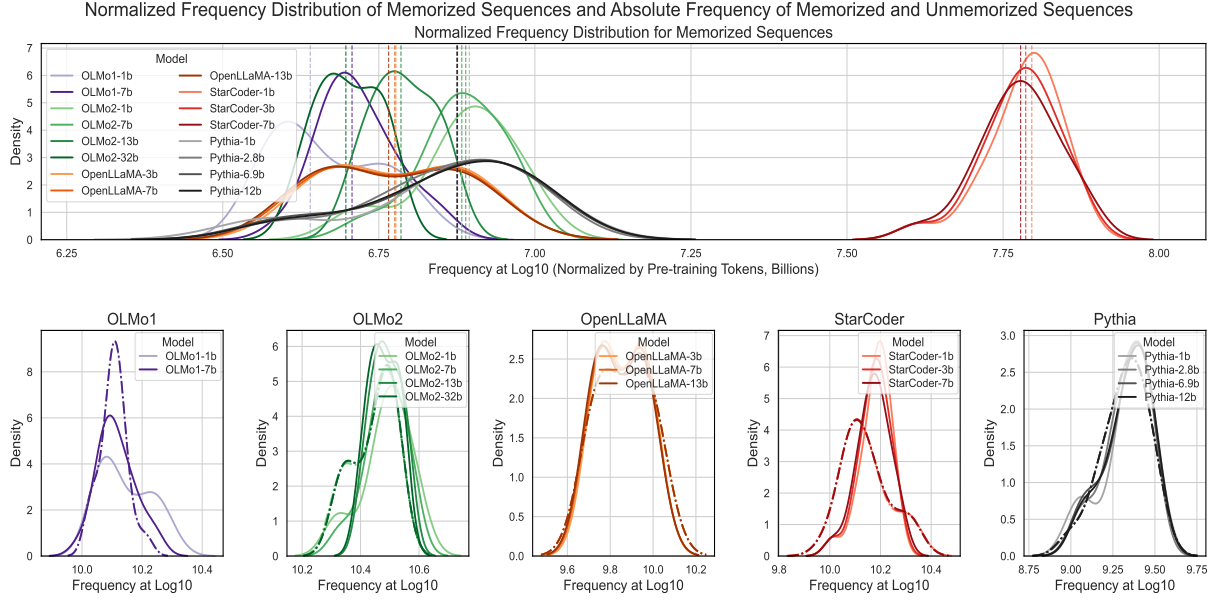
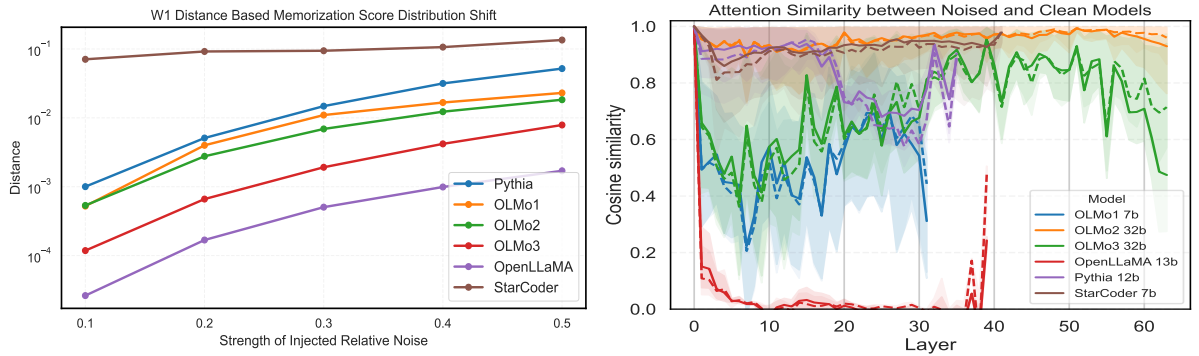


Figure 2: The average token frequency distribution for different models. The upper figure shows the normalized frequency for memorized sequences. The dotted vertical lines indicate the average frequency for each model. The lower figure shows the absolute frequency for memorized (solid line) and unmemorized (dotted line) sequences.



(a) W1 distribution distance between noised and clean models. (b) Attention similarity between noised and clean models.

Figure 3: (Left figure) Memorization score distribution shift under noise. (Right figure) The attention head similarity between clean and noised models, with the variance represented by the shaded band. Solid/dotted lines represent results for memorized/unmemorized sequences. The noise range is [0.1, 0.2, 0.3, 0.4, 0.5].

rizing certain sequences. By observing their average frequency lines, we see that these lines shift leftward as model size increases within a family when the number of training tokens is fixed, which explains why larger models memorize more sequences. From the lower figure, despite this overlap, we can see that unmemorized sequences have a higher distribution mass in the left region than memorized sequences, indicating that most unmemorized sequences tend to have a lower frequency, showing that frequency is one of the deciding factors for memorization. However, the existence of memorized low-frequency sequences and unmemorized high-frequency sequences also indicates a certain degree of randomness or an unexplored mech-

anism in memorization, showing that frequency is only one of the reasons for memorization.

4.4 Memorization Robustness Under Noise

From this section onward, we turn to the discussion of internal components, where the most intriguing question is *whether specific components exist for memorization*. Compared with previous research that studies models using perturbed input tokens (Bigoulaeva et al., 2026), we observe the change in memorization under noise perturbation of the residual stream, based on the hypothesis that if memorization relies on certain specific components, those components should be more sensitive to noise. At the statistical level, the W1 distance

is used (Givens and Shortt, 1984) to measure the memorization score distribution shift under noise. The cosine similarity of attention heads between noise and clean models is used to measure the internal change under noise.

As shown in Figure 3a, increasing noise strength shifts the memorization score distribution across all models. StarCoder is the most vulnerable to noise perturbation, as its learned representations are too domain-specific to be noise-robust. While the other models share a similar curve, for the OLMo families we see that more capable models are more robust to noise perturbation, meaning that although capable models may not have a lower compression ratio, they memorize sequences more firmly at the internal level. To understand the internal mechanics behind this robustness, we analyze the layer-wise cosine embedding similarity between the noised and clean models (Figure 3b). While noise predictably disrupts the early layers, similarity generally recovers in later layers across all models, revealing an internal error-correction mechanism. Crucially, however, memorized sequences exhibit lower similarity recovery than unmemorized sequences, indicating that memorized sequences are more sensitive to the perturbation. This suggests that LLMs possess general error-correction capabilities regardless of whether a sequence is memorized or not. However, the results show that memorized sequences are more sensitive to perturbation, suggesting the existence of specific components for memorization, i.e., if those components receive the perturbed input, information regarding the memorized sequence cannot be correctly reconstructed.

4.5 Decoding the Middle Layers

To further investigate the emergence of memorization, we apply the Logit-Lens to probe token decoding probabilities across layers (Figure 4).

For both memorized and unmemorized sequences, decoding probabilities increase gradually in the early layers before exhibiting a sharp burst in the later layers. This shared trend suggests that the late-layer probability spike is primarily a general mechanism for output generation, rather than a phenomenon exclusive to memorization. Despite this shared pattern, memorized tokens maintain a distinct probability advantage (often > 0.5) over unmemorized tokens in the later layers, becoming highly predictable well before the final layer. Linking this with our previous observation—where noise perturbation severely disrupts the internal

similarity in early layers while late layers attempt to recover it—we hypothesize a two-stage mechanism for memorization: the core features and routing for exact memorization are likely established in the early layers, whereas the later layers primarily serve to activate and translate this prepared information into final output probabilities.

4.6 Overlap of Important Heads

The results from noise robustness and layer decoding suggest that a shared pattern exists in generating memorized sequences. To locate this pattern, we use attention head ablation to trace the importance of each head for memorized examples in each domain. We then sort and compare the resulting importance-score lists across heads for each example in different domains. To ease the self-compensation effect when removing attention heads (McGrath et al., 2023; Bigoulaeva et al., 2026), instead of absolute rankings, we compare the within- and cross-domain set overlap for the top-20% most important heads in Table 3.⁵

First, the in-domain example overlap is notably high, meaning that most of the important heads are shared across examples within each domain, and this overlap increases with model size, suggesting that larger models compress domain-specific memorization into dedicated heads. Conversely, the cross-domain overlap is generally lower, meaning that the important heads of different domains do not overlap much, and this overlap decreases as models scale, indicating that heads for different domains decouple in larger parameter spaces. Despite this decoupling, for almost all models, a small but crucial subset of heads remains important for over 90% (or even all) of the domains. For example, in the OLMo3 32b model, around 1.4% of the heads are important for all domains. This indicates the existence of some heads that are crucial for memorization behavior itself.

4.7 Example Distribution of Shared Important Heads

Based on our discovery of cross-domain, universally important attention heads, we investigate their layer-wise distribution in the OLMo2-13b and Pythia-12b models, because these two models have a similar number of layers. As shown in Figure 5, these shared memorization heads are not distributed uniformly across layers but show

⁵The hypothesis test for these results and the sample coverage of the experiments are provided in Section A.10.

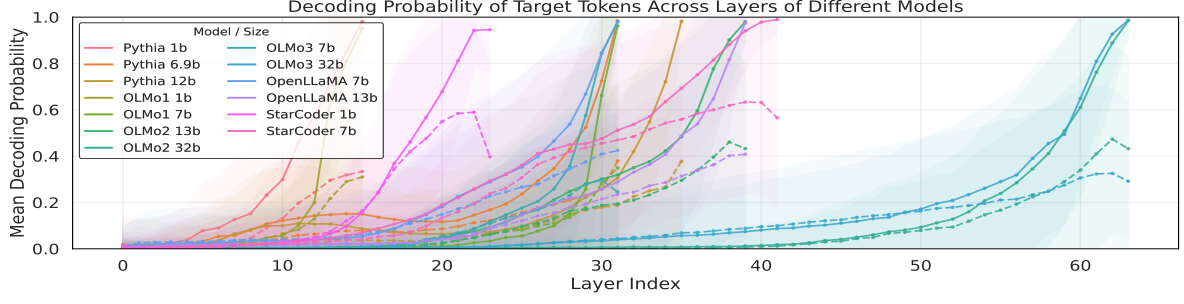


Figure 4: Decoding probability (y-axis) for memorized and unmemorized tokens across different layers (x-axis) for all models. The solid/dotted lines represent the probability for memorized/unmemorized sequences.

Model	Size	Any Two Domain Pair			In-Domain Example Pair	Percentage of Shared Important Heads Among $> x\%$ Domains				
		Mean Jaccard	Max Jaccard	Random		$> 30\% \text{ dom}$	$> 50\% \text{ dom}$	$> 70\% \text{ dom}$	$> 90\% \text{ dom}$	all dom
OLMo1	1b	28.58%	57.58%	11.39%	57.86%	28.12%	12.11%	5.08%	1.17%	0.78%
	7b	24.41%	48.01%	11.14%	80.03%	29.79%	10.84%	2.05%	0.98%	0.98%
OLMo2	1b	52.08%	89.09%	11.39%	57.69%	23.05%	18.75%	14.06%	7.42%	4.69%
	7b	21.97%	70.12%	11.14%	82.26%	23.24%	9.96%	3.42%	0.78%	0.29%
	13b	18.93%	38.23%	11.12%	87.66%	20.62%	8.94%	1.75%	0.62%	0.38%
	32b	20.76%	25.49%	11.12%	91.18%	42.58%	14.30%	3.12%	3.12%	3.12%
OLMo3	7b	22.51%	41.87%	11.14%	93.66%	20.31%	8.89%	2.64%	1.17%	0.29%
	32b	22.46%	27.05%	11.12%	96.57%	30.90%	16.17%	3.71%	1.37%	1.37%
OpenLLaMA	3b	17.21%	30.47%	11.18%	74.97%	33.29%	15.62%	1.56%	0.84%	0.84%
	7b	18.04%	33.99%	11.14%	83.91%	32.81%	15.23%	1.86%	0.49%	0.49%
	13b	17.69%	29.03%	11.12%	83.87%	31.44%	15.50%	1.44%	0.44%	0.44%
Pythia	1b	46.17%	92.59%	11.48%	58.03%	19.53%	18.75%	17.19%	1.56%	1.56%
	2.8b	17.03%	37.58%	11.14%	69.84%	16.89%	6.74%	1.76%	0.20%	0.10%
	6.9b	19.72%	39.46%	11.14%	65.98%	27.34%	11.43%	1.66%	0.39%	0.10%
	12b	14.97%	29.15%	11.13%	67.55%	16.25%	3.75%	0.76%	0.14%	0.00%
StarCoder	1b	18.38%	67.39%	11.20%	94.79%	19.01%	6.51%	2.08%	0.52%	0.26%
	3b	15.69%	24.71%	11.19%	95.29%	19.44%	4.17%	1.01%	0.00%	0.00%
	7b	13.57%	19.03%	11.14%	96.84%	19.35%	1.04%	0.00%	0.00%	0.00%

Table 3: Cross-domain top-20% head overlap against a random-selection baseline, together with the mean within-domain example-level overlap and the ratio of heads that remain in the top 20% of importance in at least X% of the domains, or in all domains.

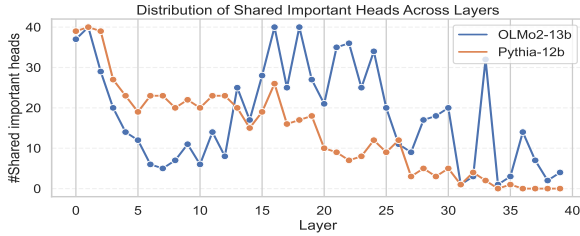


Figure 5: Distribution of shared important heads across layers for Pythia-12b and OLMo2-13b

distinct, model-dependent patterns. While the general trend is that the most important heads for both models are in the early-middle layers, the distribution after the early-middle layers differs. OLMo2 also has a certain number of important heads in the middle-to-late layers, whereas the number in Pythia decreases steadily. This fluctuating distribution further explains the results in Figure 4, where we hypothesize that the information is prepared in the early layers but activated in the later layers. In the early-middle layers, the general retrieval

for memorization relies on shared important heads. As information progresses to deeper layers, shared heads activate domain-specific memorization coming from specialized heads, leading to a decrease in shared important heads. Finally, the exact memorized tokens are activated and reinforced by the gathered information in the last layers, leading to near-deterministic decoding.

4.8 Distribution Similarity of Layer Importance Across Models

To understand whether different models share the same memorization structure or not, we compute a layer-wise importance score by taking the average of the importance scores of attention heads in each layer. This layer-level importance is normalized based on the number of layers, with distribution comparison between each model in Figure 6.

Pythia has an average similarity close to 0.9 for each model size. Notably, for OLMo1/2/3, we observe a high similarity not only within each model family but also across these three families,

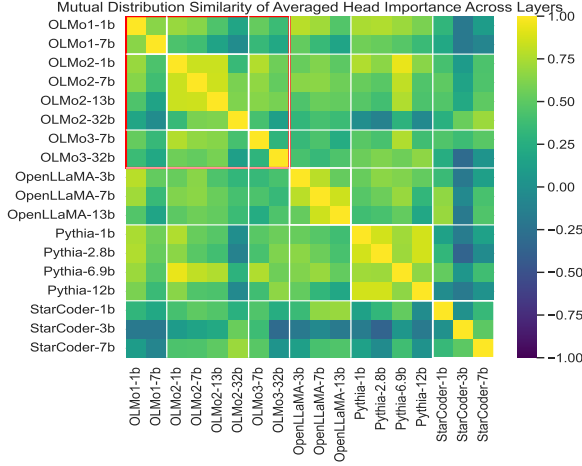


Figure 6: Distribution Similarity Heatmap of Layer-wise Average Head Importance for all models.

indicating a similar memorization structure within this successively updated model series. Additionally, different model families have a lower cross-similarity, especially StarCoder, which is the most distant from the other models. This suggests that the memorization structure of a model family is shared among its models, regardless of size. However, there is no universal memorization structure shared by all LLMs. The memorization structure of an LLM model family is determined by the training recipe (model, data, and training algorithm) of each model family.

5 Discussion

Understanding the Capacity of Memorization

While we have shown that memorization is related to model structure (model size, architecture design, etc.), data size, and data distribution, the interaction among these factors remains unexplored in this study. For example, if we could estimate the capacity of memorization given certain model training configurations, this could provide another perspective from which to inspect the training process.

Definition of Memorization We have studied verbatim memorization, but verbatim memorization is only one of the definitions used for LLMs. From a cognitive perspective, humans do not usually memorize something in a verbatim way, and human memory also relates to vague information like concepts that do not connect to a specific data fragment. The definition of memorization should also be extended to those areas, together with a practically executable measure, to help us further understand the complex behavior of LLMs.

Fine-grained Analysis of Texts of Interest Our analysis characterizes general memorization behavior across models, but many sequences are not interesting from a human perspective. It is not a concern if we find that LLMs have memorized random code fragments from the internet. However, we do care about whether a personal address, banking details, or an API key has been memorized by an LLM. Such texts, however, come from narrow domains, which means that their mechanisms or statistical behavior may exhibit unique trends beneath the general trends, and this may require further study.

6 Conclusion

In this study, we conducted a comparative analysis of LLM memorization at both the statistical and internal levels to identify cross-model commonalities and model-specific signatures. At the **statistical level**, despite individual model differences, we uncover a general log-linear scaling of memorization rate with model size, and show that memorized sequences are highly compressible. Further, the domain and frequency distributions show that large LLMs have a lower frequency threshold for memorization and tend to memorize more free text instead of the structural text. At the **internal level**, noise perturbation reveals that while LLMs show a general denoising capability, memorized sequences are more sensitive, indicating a reliance on specific components. While memorized sequences show a stronger signal, the decoding probability reveals a shared generation process. Finally, the attention ablation suggests that the memorization-important heads are shared within domains, and that a small set of them is domain-independent, indicating that they may relate to memorization itself. Crucially, the distribution of these heads is consistent within families but diverges across them, showing that the memorization structure is shaped by the model’s training recipe.

Acknowledgment

This work was partially supported by the Institute of AI and Beyond of the University of Tokyo and by the “R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models” project of the Ministry of Education, Culture, Sports, Science and Technology. This work was also supported by ABCI 3.0, which is provided by AIST and AIST Solutions.

Limitations

Although we study most of the commonly used fully open LLMs with accessible data, full coverage is impractical. Some models (e.g., Amber-7b, Apertus-70b)^{6 7} release only their pre-training data collection scripts rather than the data itself, and re-running these scripts rarely reproduces the original data because of broken links, whereas studying memorization requires the original corpora. Most open-weight models (Qwen, DeepSeek, Llama, etc.) withhold their pre-training data. However, we kindly think this does not limit our findings only to data-transparent models. These models typically combine proprietary in-house data with publicly crawled corpora, which also overlap substantially with the corpora of data-transparent models. Additionally, these models are also built on frontier research results, although their individual structures may differ. Data-transparent models are also competitive with mainstream open-weight ones (OLMo3 vs. Qwen 2.5/3). As our experiments are cross-model, we believe that similar results would also hold for open-weight models.

Infini-gram is the only service supporting token-frequency queries over popular pre-training corpora, but its uniform Llama-2 tokenization differs from the native tokenizers of the models we study, potentially skewing the measured frequencies. We accept this trade-off because no alternative exists—building a fast-search API over hundreds of terabytes of data is itself a substantial engineering challenge—and because our analysis relies on sequence-level average frequencies rather than on individual token counts, an approach that is more robust to such skew. Several findings also align with prior work, offering indirect validation.

Pre-training corpora are too large to analyze exhaustively at reasonable cost, so our results are based on samples. The consistent trends we observe across model families, sizes, and domains suggest the sampling is sufficient to surface general patterns.

Ethical Considerations

The use of AI tools for this paper is limited to proofreading; for the research as a whole, AI was also used for coding assistance, and the results were validated by the authors.

While we have analyzed the sampled pre-training data and inspected some examples for observation, we did not observe any offensive language or information related to personal privacy. Additionally, when sampling the Pile pre-training data, we sampled from the version from which content with copyright issues has been removed⁸. These domains include Books3, BookCorpus2, OpenSubtitles, YTSsubtitles, and OWT2, and we did not sample data from them.

For all models and data used in this study, we align our usage with their license and intended usage.

References

- Anthropic. 2025. [Claude Sonnet 4.5 system card](#).
- Stefan Arnold. 2025. [Memorization in language models through the lens of intrinsic dimension](#). In *Proceedings of the First Workshop on Large Language Model Memorization (L2M2)*, pages 23–28, Vienna, Austria. Association for Computational Linguistics.
- Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2025. [Eliciting latent predictions from transformers with the tuned lens](#). *Preprint*, arXiv:2303.08112.
- Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023a. Emergent and predictable memorization in large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Stella Biderman, Hailey Schoelkopf, Quentin 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, ICML'23*. JMLR.org.
- Irina Bigoulaeva, Jonas Rohweder, Subhabrata Dutta, and Iryna Gurevych. 2026. [Patches of nonlinearity: Instruction vectors in large language models](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12209–12262, San Diego, California, United States. Association for Computational Linguistics.

⁶Amber Data Script

⁷Apertus Data Script

⁸Hugging Face Link for Pile Uncopyrighted

- Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. [Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow](#).
- Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. [Quantifying memorization across neural language models](#). In *The Eleventh International Conference on Learning Representations*.
- 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.
- Anton Changalidis and Aki Härmä. 2025. [Capacity matters: a proof-of-concept for transformer memorization on real-world data](#). In *Proceedings of the First Workshop on Large Language Model Memorization (L2M2)*, pages 227–238, Vienna, Austria. Association for Computational Linguistics.
- Bowen Chen, Namgi Han, and Yusuke Miyao. 2024. [A multi-perspective analysis of memorization in large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11190–11209, Miami, Florida, USA. Association for Computational Linguistics.
- Kang Chen, Xiuze Zhou, Yuanguo Lin, Shibo Feng, Li Shen, and Pengcheng Wu. 2025. [A survey on privacy risks and protection in large language models](#). *Preprint*, arXiv:2505.01976.
- A. Feder Cooper, Mark A. Lemley, Allison Casasola, Ahmed Ahmed, Aaron Gokaslan, Amy B. Cyphert, Christopher De Sa, Daniel E. Ho, and Percy Liang. 2026. [Extracting memorized pieces of \(copyrighted\) books from open-weight language models](#). *Preprint*, arXiv:2505.12546.
- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. [Knowledge neurons in pretrained transformers](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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](#). *Preprint*, arXiv:2101.00027.
- Xinyang Geng and Hao Liu. 2023. [Openllama: An open reproduction of llama](#).
- Clark R. Givens and Rae Michael Shortt. 1984. [A class of Wasserstein metrics for probability distributions](#). *Michigan Mathematical Journal*, 31(2):231 – 240.
- Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. [OLMo: Accelerating the science of language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Valentin Hartmann, Anshuman Suri, Vincent Bindschadler, David Evans, Shruti Tople, and Robert West. 2023. [Sok: Memorization in general-purpose large language models](#). *Preprint*, arXiv:2310.18362.
- Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Iliia Shumailov, Milad Nasr, Christopher A. Choquette-Choo, Katherine Lee, and A. Feder Cooper. 2025. [Measuring memorization in language models via probabilistic extraction](#). 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 9266–9291, Albuquerque, New Mexico. Association for Computational Linguistics.
- Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, and Nicholas Carlini. 2023. [Preventing generation of verbatim memorization in language models gives a false sense of privacy](#). In *Proceedings of the 16th International Natural Language Generation Conference*, pages 28–53, Prague, Czechia. Association for Computational Linguistics.
- Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. [The stack: 3 tb of permissively licensed source code](#). *Preprint*, arXiv:2211.15533.
- Ilya Lasy, Peter Knees, and Stefan Woltran. 2025. [Understanding verbatim memorization in LLMs through circuit discovery](#). In *Proceedings of the First*

- Workshop on Large Language Model Memorization (L2M2)*, pages 83–94, Vienna, Austria. Association for Computational Linguistics.
- Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: a case study on dpo and toxicity. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. [Deduplicating training data makes language models better](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.
- Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, and 39 others. 2023. [Starcoder: may the source be with you!](#) *Transactions on Machine Learning Research*. Reproducibility Certification.
- 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*.
- Xingyu Lu, Xiaonan Li, Qinyuan Cheng, Kai Ding, Xuanjing Huang, and Xipeng Qiu. 2024. [Scaling laws for fact memorization of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11263–11282, Miami, Florida, USA. Association for Computational Linguistics.
- Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Beguelin. 2023. [Analyzing Leakage of Personally Identifiable Information in Language Models](#). In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363, Los Alamitos, CA, USA. IEEE Computer Society.
- Xiaoyu Luo, Yiyi Chen, Johannes Bjerva, and Qiongxiu Li. 2025. [Shared path: Unraveling memorization in multilingual LLMs through language similarities](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19372–19388, Suzhou, China. Association for Computational Linguistics.
- Iris Ma, Ian Domingo, Alberto Krone-Martins, Pierre Baldi, and Cristina Lopes. 2025. [Memorization: A close look at books](#). In *Proceedings of the First Workshop on Large Language Model Memorization (L2M2)*, pages 169–182, Vienna, Austria. Association for Computational Linguistics.
- Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. 2023. [The hydra effect: Emergent self-repair in language model computations](#). *Preprint*, arXiv:2307.15771.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA. Curran Associates Inc.
- nostalgebraist. 2020. [interpreting GPT: the logit lens](#). LessWrong.
- Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, and 50 others. 2026. [Olmo 3](#). *Preprint*, arXiv:2512.13961.
- Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 24 others. 2025. [2 olmo 2 furious](#). *Preprint*, arXiv:2501.00656.
- OpenAI. 2025. [GPT-5 system card](#).
- Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary C. Lipton, and J. Zico Kolter. 2024. Rethinking llm memorization through the lens of adversarial compression. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS ’24*, Red Hook, NY, USA. Curran Associates Inc.
- Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxu Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. [Dolma: an open corpus of three trillion tokens for language model pretraining research](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.
- Hiromu Takahashi and Shotaro Ishihara. 2025. [Quantifying memorization in continual pre-training with Japanese general or industry-specific corpora](#). In *Proceedings of the First Workshop on Large Language Model Memorization (L2M2)*, pages 95–105, Vienna, Austria. Association for Computational Linguistics.
- Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus,

- Tristan Hume, Francesco Mosconi, C. Daniel Freeman, and 7 others. 2026. [Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet](#). *Preprint*, arXiv:2605.29358.
- Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: analyzing the training dynamics of large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024. [Knowledge editing for large language models: A survey](#). *ACM Comput. Surv.*, 57(3).
- Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA. Curran Associates Inc.
- Alexander Xiong, Xuandong Zhao, Aneesh Pappu, and Dawn Song. 2025. [The landscape of memorization in llms: Mechanisms, measurement, and mitigation](#). *Preprint*, arXiv:2507.05578.
- Wenli Yang, Lilian Some, Michael Bain, and Byeong Kang. 2025. [A comprehensive survey on integrating large language models with knowledge-based methods](#). *Knowledge-Based Systems*, 318:113503.
- Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. [Back attention: Understanding and enhancing multi-hop reasoning in large language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11257–11272, Suzhou, China. Association for Computational Linguistics.
- Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Counterfactual memorization in neural language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Danyang Zhang, Junhao Song, Ziqian Bi, Xinyuan Song, Yingfang Yuan, Tianyang Wang, Joe Yeong, and Junfeng Hao. 2025. [Mixture of experts in large language models](#). *Preprint*, arXiv:2507.11181.
- Xuanhe Zhou, Junxuan He, Wei Zhou, Haodong Chen, Zirui Tang, Haoyu Zhao, Xin Tong, Guoliang Li, Youmin Chen, Jun Zhou, Zhaojun Sun, Binyuan Hui, Shuo Wang, Conghui He, Zhiyuan Liu, Jingren Zhou, and Fan Wu. 2025. [A survey of llm \$\times\$ data](#). *Preprint*, arXiv:2505.18458.

A Appendix

A.1 Experiment Details

A.1.1 Device and Running Settings

The computing devices used in this research are mixed. We have used H100, H200, A100, and A6000 servers. In this study, we did not train any model; all CUDA devices were used only for generation and internal probing. The running time depends on the model and its size. Generating all memorization scores across six noise strengths for all models used in this study took around two months on eight A100 servers.

For the compression ratio analysis, it took around two weeks to run the experiments on three A6000 devices.

For the middle layer decoding analysis, it took around two weeks to run the experiments on eight A100 devices.

For the head importance results, it took around one month to run all model sizes across their domains on an H100 device. The total number of attention heads grows very quickly as model size increases, since model size scaling is usually accompanied by an increase in the number of layers and an expansion of head size, which means that increasing the model size leads to experiments on hundreds of additional attention heads. In particular, we need to evaluate each head and regenerate the sequence without this head, which means that a single sequence requires hundreds of additional generations when we use a larger model in a model family. Therefore, we did not conduct experiments on all memorized examples in the head ablation study, as the computational cost is not affordable in practice. For small models (1b to 7b), we sample 10,000 memorized examples in each domain. For large models (above 7b), we sample 2,500 memorized examples in each domain.

A.1.2 Data Details

We present the data domains for each model and their corresponding classifications into the Free-Text, Semi-Structural, and Structural text categories in Table 4.

For the collection of the data, we referred to either their released self-collected HuggingFace dataset⁹ or the dataset that was clearly mentioned in their posts or repository¹⁰. We use a verified

⁹dolma data, ,olmo-mix-1124 data,dolmino-mix-1124 data ,dolma3 data

¹⁰the stack data

token to access that data, download it to the server, and split it by domain. After collecting the data, we pre-process it by passing the documents to the tokenizer of the corresponding target model and filtering out the documents that have fewer than 128 tokens; we note that only a very small amount of data is filtered out by this process.

A.1.3 Obtaining Token Frequency

In this study, we have used Infini-gram¹¹ to obtain the token frequencies of memorized and unmemorized sequences. We employ its official Python API to obtain the frequencies.

We also note that the limitations of our frequency analysis stem from Infini-gram. First, Infini-gram mainly uses the Llama-2 tokenizer for its n-gram database. In contrast, we use the corresponding tokenizer of each model in our other experiments, for example, the Pythia tokenizer for the Pythia models. Furthermore, based on our observations, the Llama-2 tokenizer tends to give shorter tokenized lists than the tokenizer of the corresponding original model. For example, we find that some sequences with 64 tokens in the OLMo2 models have 46 tokens under the Llama-2 tokenizer. Therefore, we note that the results of our frequency analysis are approximate.

Second, due to the API limit of Infini-gram, we queried the frequency only for fully memorized, half-memorized, and unmemorized sequences.

Third, Infini-gram does not provide a Python API endpoint for the OLMo3 data. This is why we do not report the frequency results for the OLMo3 models.

A.2 Detailed Distribution Over Domains

In this section, we show the detailed distribution and exact counts for each domain for all models in Table 7.

In this table, we observe essentially the same results, where most of the memorized documents come from code and the math domain, followed by arXiv and Wikipedia, and finally by free-text domains such as PES20 or OCRed documents. However, some domain classifications are also coarse-grained, such as Pile-CC in Pythia and flan in the OLMo2 models, which may contain documents

of various types, making a very accurate domain-specific distribution hard to obtain.

For the StarCoder models, since all training data consists of code in different languages, we observe a much less polarized distribution of memorized documents than in the other models. However, we still observe a very consistent ordering of proportions across model sizes. Additionally, this ordering does not seem to strictly relate to the popularity of programming languages, as Python is only in the middle of the ranking, while Scala, which is famous for its variable expressions, has the largest share. Nonetheless, the trend still generally correlates with the popularity and grammaticality of the corresponding languages, since Java, C, HTML, Go, and Kotlin have the highest shares, and these languages are either common (C, HTML, Java) or very predictable (Go, Kotlin), whereas less popular languages (Ruby, R, Julia) have the smallest shares.

A.3 Domain Distribution with Injected Noise

In this section, we also discuss the distribution over the three domain categories (Structural, Semi-Structural, and Free-Text) under different injected relative noise strengths (0.1, 0.3, and 0.5) in Table 5.

From the results, we can see that the distributions of Pythia and OLMo1 are more affected than those of the other models. When increasing the strength of the noise, the distribution largely shifts to the Structural domain, where OLMo1 7b’s Structural domain increased from 51.3% to 91.3% when changing the noise to 0.5. Similar results are also observed for the Pythia models, where the Structural proportion increases from 74.4% to 91.1% for the 12b model. However, the same tendency is weaker (OpenLLaMA) or not observable (OLMo2/3) in the other models, where the distribution barely changes. In particular, the distribution of OLMo3 is essentially unaffected by the increase in injected noise strength. The reason may be that more capable models memorize free text through semantic information, which is as robust as the code or mathematical expressions memorized through repetitive patterns.

We can observe that injecting stronger noise shifts the curves downward, meaning that the distribution moves to the left region, which is expected. However, we do observe that the OLMo1 models are largely affected by the injected noise, where the highest memorization rate of OLMo1 7b decreased from near 0.1 to 0.001, which is almost

pile data
redpajama data
¹¹<https://infini-gram.readthedocs.io/en/latest/api.html>

Model Family	Domain Type	Domains	#Dom
Pythia	Structural	USPTO_Backgrounds, EuroParl, Enron_Emails, Github, StackExchange	5
	Semi-Structural	PubMed_Abstacts, PubMed_Central, FreeLaw, NIH_ExPorter	4
	Free-Text	arXiv, DM_Mathematics, Gutenberg (PG-19), HackerNews, Pile-CC, PhilPapers, Ubuntu_IRC, Wikipedia (en)	8
OLMo1	Structural	starcode, proof_pile_2_algebraic_stack, proof_pile_2_open_web_math	3
	Semi-Structural	redpajama_stackexchange, cc_news_head, cc_news_middle, cc_news_tail, c4_filtered, cc_en_head, cc_en_middle, cc_en_tail	8
	Free-Text	books, falcon_refinedweb_filtered, reddit, redpajama_arxiv, wiki, pes2o, tulu_flan, wikiref_megawika	8
OLMo2	Structural	stackexchange, open-web-math, starcode, math	4
	Semi-Structural	arxiv, pre_train_wiki, pre_train_dclm, pre_train_pes2o, pes2o	5
	Free-Text	default, flan, dclm	3
OLMo3	Structural	stack_edu_C, stack_edu_CSharp, stack_edu_Cpp, stack_edu_Go, stack_edu_Java, stack_edu_JavaScript, stack_edu_Markdown, stack_edu_PHP, stack_edu_Python, stack_edu_Ruby, stack_edu_Rust, stack_edu_SQL, stack_edu_Shell, stack_edu_Swift, stack_edu_TypeScript	15
	Semi-Structural	dolma1_7_wiki_en, finemath_3plus, rpj_proofpile_arxiv, olmocr_science_pdfs_adult_content, olmocr_science_pdfs_art_and_design, olmocr_science_pdfs_education_and_jobs, olmocr_science_pdfs_entertainment, olmocr_science_pdfs_sports_and_fitness	9
	Free-Text	common_crawl_adult_content, common_crawl_art_and_design, common_crawl_crime_and_law, common_crawl_education_and_jobs, common_crawl_electronics_and_hardware, common_crawl_entertainment, common_crawl_fashion_and_beauty, common_crawl_finance_and_business, common_crawl_food_and_dining, common_crawl_games, common_crawl_health, common_crawl_history_and_geography, common_crawl_home_and_hobbies, common_crawl_industrial, common_crawl_literature, common_crawl_politics, common_crawl_religion, common_crawl_science_math_and_technology, common_crawl_social_life, common_crawl_software, common_crawl_software_development, common_crawl_sports_and_fitness, common_crawl_transportation, common_crawl_travel_and_tourism	24
OpenLLaMA	Structural	github_sample, stackexchange_sample	2
	Semi-Structural	arxiv_sample, wikipedia_sample	2
	Free-Text	c4_sample, common_crawl_sample	2
StarCoder	Structural	c, c_sharp, css, dockerfile, go, html, java, javascript, julia, kotlin, lua, php, python, r, ruby, rust, scala, shell, sql, swift, typescript	21

Table 4: Complete domain classification and counts by model family. Domains are categorized as Structural (code/mathematics), Semi-Structural (mixed format), and Free-Text (natural language prose). The #Dom column shows the count of domains in each category.

100 times lower than the non-noised model. Additionally, in most cases, larger models are more robust compared to their small counterparts, where we also observed that the high memorization score probability density of OLMo2 1b decreases significantly, while larger models like 32b only show very limited influence caused by the noise injection. We also notice that the distribution of those lines is more dispersed at higher noise levels than at low noise levels. This means that the decreasing trend is not the same for all model families, showing that robustness to the injected noise is a model-specific feature rather than a common feature that is shared within a model family. The reason may be differences in model parameters and in training parameters that change with model size.

A.4 Similarity for the MLP Layer

In this section, we show the similarity between the noised and clean models for the MLP layer in Figure 7.

In this figure, we see that the MLP layer shows essentially a trend similar to that of the attention heads. The similarity also decreases in the initial layers, then bounces back in later layers. The similarity of memorized sequences is also lower in the later layers than that of unmemorized sequences. We also notice that, for memorized sequences, the similarity to the original model is higher at early layers compared to unmemorized sequences. This means that the information itself is strong in the early layers, but that more specific information is required for those memorized sequences to be successfully decoded. It means those memorized sequences provide stronger information clues than

Model	Pythia			OLMo1		OLMo2			OLMo3		OpenLLaMA		
Size	1b	2.8b	12b	1b	7b	7b	13b	32b	7b	32b	3b	7b	13b
Structural	75.2%	76.4%	74.4%	58.3%	51.3%	44.9%	44.0%	39.8%	49.0%	44.4%	66.1%	61.5%	58.6%
Semi-Structural	19.6%	17.4%	17.2%	31.1%	33.5%	9.5%	10.5%	12.1%	6.4%	7.9%	12.0%	13.5%	13.1%
Free-Text	5.1%	6.2%	8.4%	10.6%	15.2%	45.6%	45.5%	48.1%	44.6%	47.7%	21.9%	25.1%	28.3%

(a) Noise = 0.1

Model	Pythia			OLMo1		OLMo2			OLMo3		OpenLLaMA		
Size	1b	2.8b	12b	1b	7b	7b	13b	32b	7b	32b	3b	7b	13b
Structural	86.0%	80.0%	79.6%	73.5%	86.6%	48.2%	44.3%	40.0%	49.8%	44.5%	67.0%	61.6%	58.6%
Semi-Structural	11.4%	16.5%	15.2%	19.6%	5.5%	10.0%	10.5%	12.2%	6.4%	7.8%	11.4%	13.4%	13.1%
Free-Text	2.6%	3.4%	5.2%	6.9%	7.9%	41.8%	45.2%	47.9%	43.8%	47.7%	21.6%	25.0%	28.3%

(b) Noise = 0.3

Model	Pythia			OLMo1		OLMo2			OLMo3		OpenLLaMA		
Size	1b	2.8b	12b	1b	7b	7b	13b	32b	7b	32b	3b	7b	13b
Structural	86.6%	91.6%	91.1%	85.3%	91.3%	55.9%	45.2%	40.5%	54.2%	44.6%	70.2%	61.8%	58.6%
Semi-Structural	8.5%	5.9%	5.2%	7.4%	3.3%	9.5%	10.5%	12.3%	6.2%	7.7%	9.1%	13.6%	13.1%
Free-Text	4.9%	2.5%	3.8%	7.3%	5.4%	34.6%	44.2%	47.2%	39.6%	47.7%	20.7%	24.7%	28.3%

(c) Noise = 0.5

Table 5: Proportion of memorized texts (score=1) across Structural, Semi-Structural, and Free-Text categories at different noise levels.

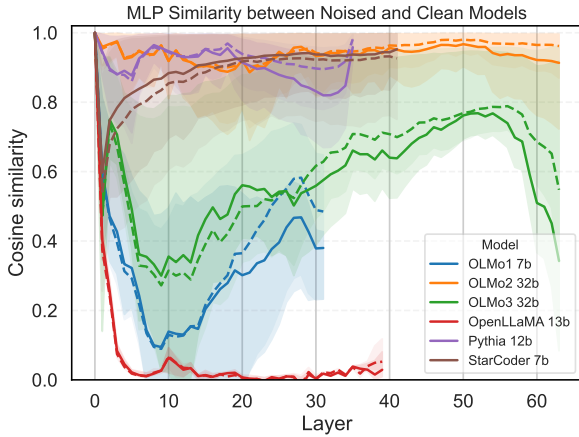


Figure 7: MLP similarity between noised and clean models. Solid/dotted lines represent the experimental results for memorized/unmemoirized sequences.

unmemorized sequences, but decoding them would require very accurate information; in a sense, little perturbed information would drive the LLMs not to generate memorized sequences even though those memorized sequences are highly correlated with model parameters.

Through comparison with the results of the attention head similarity, we also notice that the results are smoother, indicating the internal robustness of each head differs within each layer. Additionally, we can also observe that the MLP layer tends to have a lower similarity than the attention heads. This may be because the MLP layer aggregates the outputs of all attention heads, so inconsistent sim-

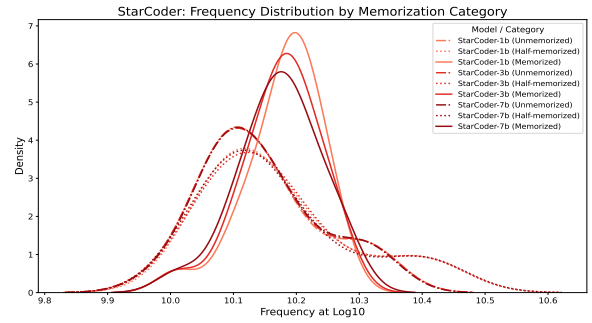
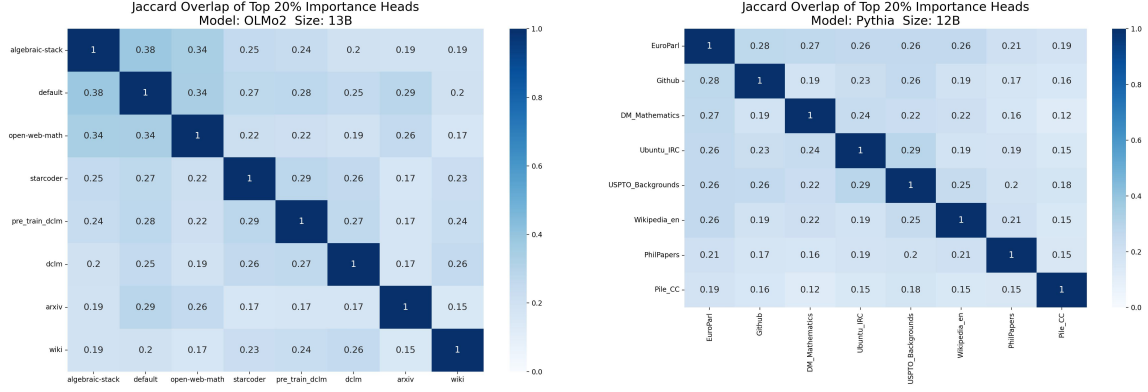


Figure 8: Frequency distribution for memorized, half-memorized, and unmemoirized sequences.

ilarity across attention heads can also reduce the similarity of the MLP layer to the original model.

A.5 Examples for Frequency Distribution Including Half-Memorized Sequences

In this section, we present distributions that also include the half-memorized sequences for the StarCoder model family in Figure 8. From these results, we can observe that the frequency of half-memorized sequences lies between those of the fully memorized and the unmemoirized sequences, since the distribution area on the left side is smaller than that of the unmemoirized sequences but larger than that of the memorized sequences. This indicates, generally, that a higher frequency leads to more memorized sequences.



(a) Top 20% important heads overlap for OLMo2 13b models. (b) Top 20% important heads overlap for Pythia 12b models.

Figure 9: Top 20% important heads overlap for Pythia and OLMo2 models on selected domains.

A.6 Examples of Shared Important Head Overlap Across Domains

In this section, we show two examples of the overlap ratio between two domains for the Pythia 12b and OLMo2 13b models. The results are presented in Figure 9.

From this result, we can observe that, for any selected domains in those two models, they share around 20% overlap in important attention heads, which aligns with the results listed in Table 3. In the left figure, we see that algebraic-stack has a higher overlap with open-web-math. The reason may be that both domains focus on math-related content. In the right figure, EuroParl has a higher overlap with other domains in general, and we also notice that it is especially close to GitHub and DM Mathematics, reaching around 30% overlap. The reason may be that EuroParl contains mainly non-English documents, which are not memorized through semantic understanding but through hard-coded memorization. This makes the memorization pattern of EuroParl closer to that of domains that are memorized with hard rules, such as code or math.

A.7 Averaged Layer Importance of All Models

In this section, we present the normalized layer importance for different model families in Figure 12 to show the individual layer-level importance structure.

From this figure, we can observe that the normalized distributions within each model family share certain common features. For example, for the Pythia models, the layer importance drops across layers for all sizes from 1b to 12b. Additionally, OLMo1 also presents a similar behavior, where the

importance score gradually decreases and then increases in the middle layers. Similar patterns are also observed in other model families. For example, even for StarCoder, where no similar trend can be observed, all models show a very unstable layer importance score, regardless of model size. The reason may be that a domain-specific model has a very different internal representation since the learned knowledge is limited to one domain, so the domain-specific knowledge is distributed across the model rather than being constrained in a fixed pattern. However, when we look at those curves across models, we can hardly identify common features, except that the beginning layers generally have a high importance score.

A.8 Examples of Unnormalized Average Layer Importance

We also provide a specific example of the average layer importance of the OLMo1 models in Figure 10, which is not normalized and serves as an example distribution. From this result, we can see that the average layer importance decreases as model size scales, which is expected since the number of heads also increases with model size. For the OLMo1 1b model, removing an attention head at layer 0 gives an importance score of 0.15, meaning that 15% of the output tokens change compared with the original output. While at the OLMo1 7b model, removing one head at layer 0 gives around a 0.05 importance score, indicating only 5% of output tokens changed compared to the original output. This change is almost one-third of that of the 1b model.

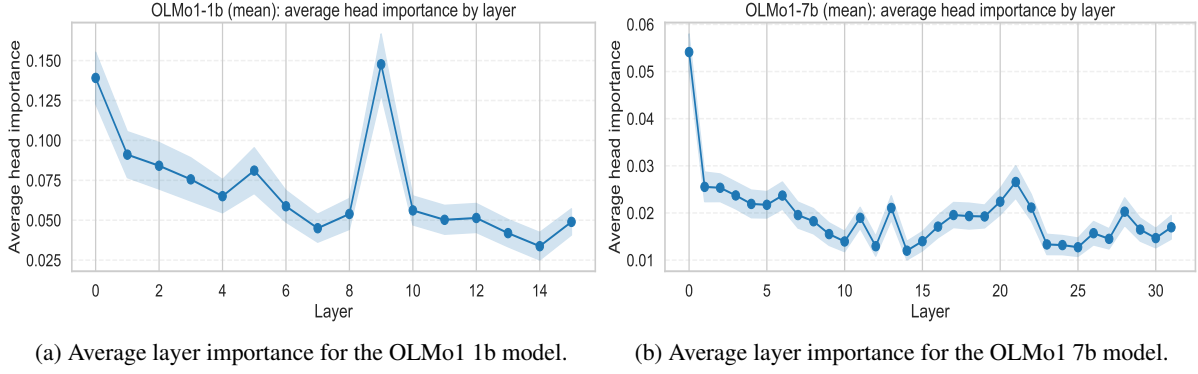


Figure 10: Average layer importance across layers.

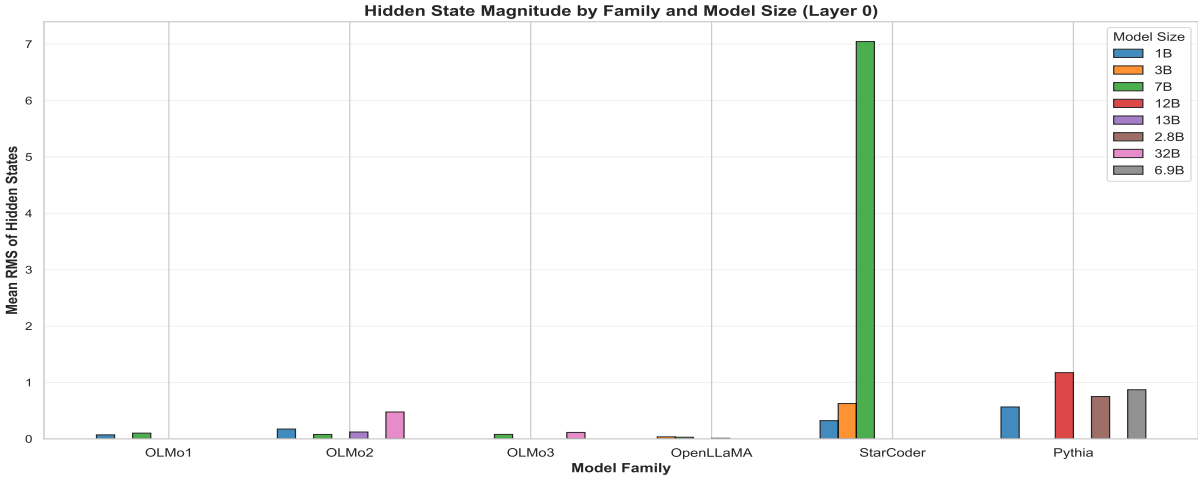


Figure 11: Average norm value of the residual stream for different models.

A.9 Absolute Residual Stream Value

In this study, the injected noise is set relative to the norm value of the residual stream. However, as different models have different MLP structures and training processes, the exact norm value also differs across model families. In this section, we present the norm value of the residual stream at the first layer as reference material for the relative noise injection. The results are presented in Figure 11. From the results, we can see that the residual stream of different models differs a lot. The StarCoder and Pythia families have a higher residual stream value, especially the StarCoder 7b, which shows a significant burst at the 7b models. Their relatively larger values also lead to a large absolute injected noise, which explains why they are less robust to noise perturbation. This could also be seen from the OpenLLaMA model, which has the lowest average residual stream norm. This also makes this model family the most robust to noise, as reflected in its memorization distribution shift. The OLMo1/2/3 families show a similar value range, showing the

effect brought by their inherited training recipe.

A.10 Significance Test for Attention Head Ablation and Domain Coverage for Memorized Sequences

To provide supporting evidence that our sampled memorized sequences justify the results obtained in this study, we report two additional analyses in Table 6: a significance test for the important-head overlap experiment, and the domain coverage of the sampled memorized sequences. The significance test establishes that the overlap of important memorization heads across models exceeds what would be expected by chance, and the coverage statistics show that our sampled examples span a wide range of domains, supporting the generality of our findings.

Significance of the head overlap study based on attention head ablation. For each pair of models, we compute the Jaccard overlap between their sets of important memorization heads, and compare it against the overlap expected when heads are

Model	Size	Head Overlap (Jaccard)				Domain Coverage of Sampled Memorized Sequences						
		Mean	Max	Rand.	Med. p	Smp./Dom.	#Dom.	Mem.	Cov.	Full Dom.	Avg Cov.	
OLMo1	1b	28.58	57.58	11.39	$3.4e-5$	10,000	19	5,785	100.0	19 / 19	100.0	
	7b	24.41	48.01	11.14	$2.8e-11$	10,000	19	11,287	100.0	19 / 19	100.0	
OLMo2	1b	52.08	89.09	11.39	$7.4e-18$	10,000	14	39,219	94.2	13 / 14	98.7	
	7b	21.97	70.12	11.14	$2.3e-6$	10,000	14	60,509	85.9	12 / 14	95.7	
	13b	18.93	38.23	11.12	$1.6e-5$	2,500	14	69,080	36.1	6 / 14	69.3	
	32b	20.76	25.49	11.12	$4.2e-14$	2,500	14	96,284	29.3	4 / 14	57.3	
OLMo3	7b	22.51	41.87	11.14	$8.8e-10$	10,000	47	164,336	100.0	47 / 47	100.0	
	32b	22.46	27.05	11.12	$3.6e-25$	2,500	47	244,693	42.8	7 / 47	56.3	
OpenLLaMA	3b	17.21	30.47	11.18	$1.7e-3$	10,000	6	22,855	78.0	5 / 6	94.4	
	7b	18.04	33.99	11.14	$5.0e-4$	10,000	6	26,681	76.4	5 / 6	93.6	
	13b	17.69	29.03	11.12	$8.1e-6$	2,500	6	31,365	35.1	2 / 6	69.3	
Pythia	1b	46.17	92.59	11.48	$1.2e-7$	10,000	13	8,368	100.0	13 / 13	100.0	
	2.8b	17.03	37.58	11.14	$5.0e-4$	10,000	14	9,147	100.0	14 / 14	100.0	
	6.9b	19.72	39.46	11.14	$5.4e-6$	10,000	15	10,531	100.0	15 / 15	100.0	
	12b	14.97	29.15	11.13	$5.1e-3$	2,500	16	10,647	44.8	15 / 16	95.6	
StarCoder	1b	18.38	67.39	11.20	$1.4e-2$	10,000	21	426,831	42.4	10 / 21	69.2	
	3b	15.69	24.71	11.19	$1.9e-2$	10,000	21	474,193	39.1	6 / 21	67.3	
	7b	13.57	19.03	11.14	$5.2e-2$	10,000	21	524,708	36.3	4 / 21	60.4	
Group 1: 1b–7b (10,000 smp./dom.), 13 models									80.6		90.6	
Group 2: $\geq 12b$ (2,500 smp./dom.), 5 models									37.6		69.6	
Overall, 18 models									68.9		84.8	

Table 6: Head overlap statistics and domain coverage of the sampled memorized sequences. All percentage columns are in %. *Rand.* is the expected Jaccard overlap under a random-sampling baseline; *Med. p* is the median p -value over all model pairs. *Smp./Dom.* is the per-domain sampling budget, *Mem.* the total number of memorized sequences, *Cov.* the ratio of sampled to memorized sequences, and *Avg Cov.* that ratio averaged over domains. The bottom block groups models by sampling budget.

drawn uniformly at random from the same head budget. The random baseline is stable at roughly 11% across all models, reflecting the fact that the number of selected heads relative to the total is comparable across architectures. Every model exceeds this baseline: mean overlap ranges from 13.57% (StarCoder 7b) to 52.08% (OLMo2 1b), so the observed agreement is not an artifact of the selection budget.

The median pairwise p -value is below 0.05 for all but one configuration. The strongest effects appear in OLMo3 32b ($p = 3.6e-25$), OLMo2 1b ($p = 7.4e-18$) and OLMo2 32b ($p = 4.2e-14$), indicating that for these models the majority of pairwise comparisons are significant by a wide margin. The single exception is StarCoder 7b, whose median p of 0.052 falls marginally short of the conventional threshold; its mean overlap of 13.57% is also the closest to the random baseline in our study. We therefore treat the head-overlap result as well supported in general, while noting that it is weakest for the largest code-only model.

Domain coverage. For each domain we sample at most 10,000 or 2,500 memorized examples, depending on the memorization rate of the model. This cap is necessary because the attention head ablation study regenerates every memorized sequence

once per ablated head. The cost therefore grows in three directions simultaneously as model size increases: larger models memorize more sequences, and they have both more layers and more attention heads per layer. Without a sampling cap, running the ablation on the largest models would be infeasible within our computation budget.

Table 6 reports the resulting coverage. Under this strategy, we cover 80.6% of all memorized sequences for small models and 37.6% for large models. Measured per domain, coverage is 90.6% for small models and 69.6% for large models, so the ablation study examines more than half of the memorized examples in each domain regardless of model scale.

Layer-profile shape comparison within model families

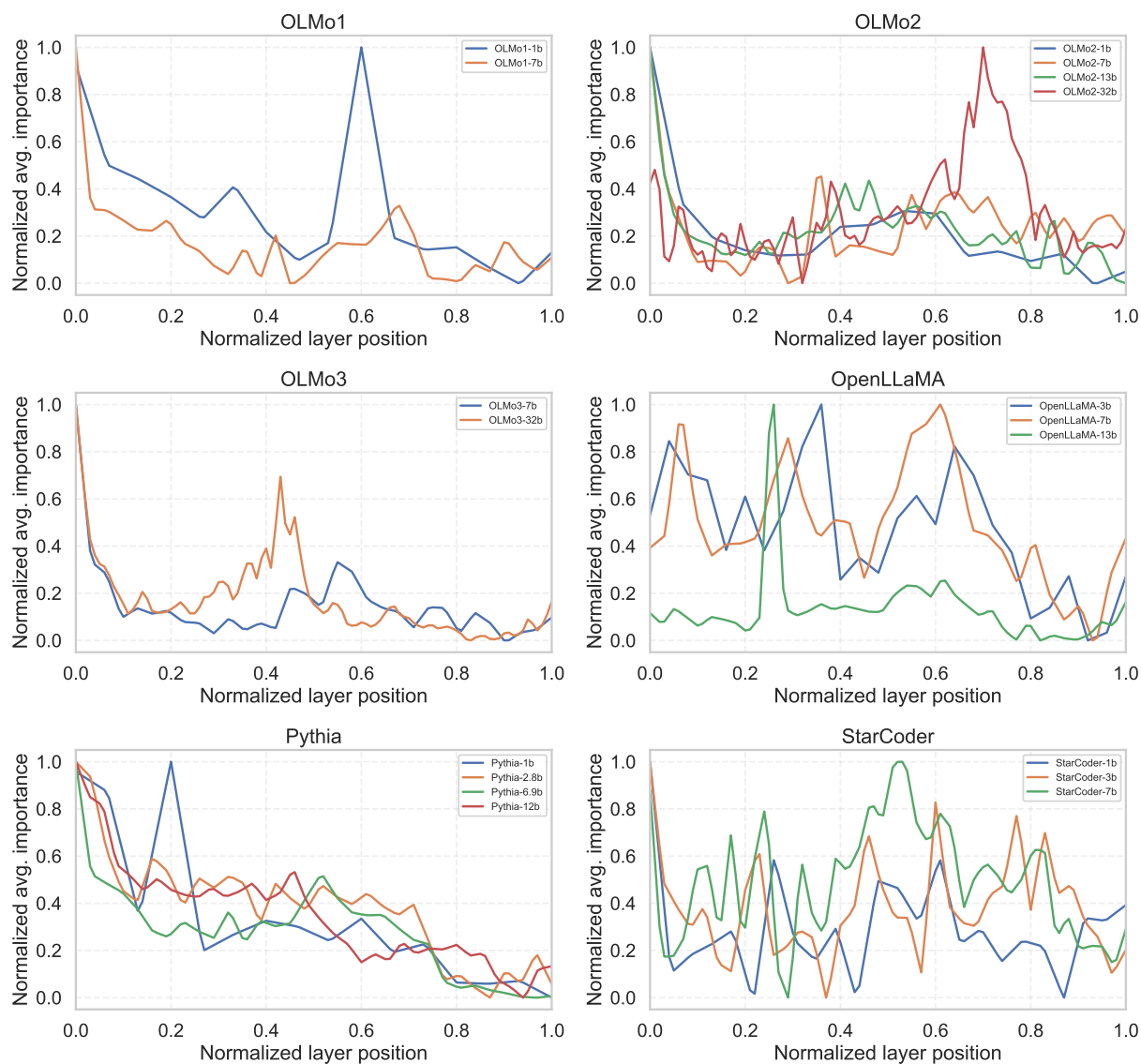


Figure 12: Layer-Normalized Averaged Importance Score for Attention Heads Across Layers.

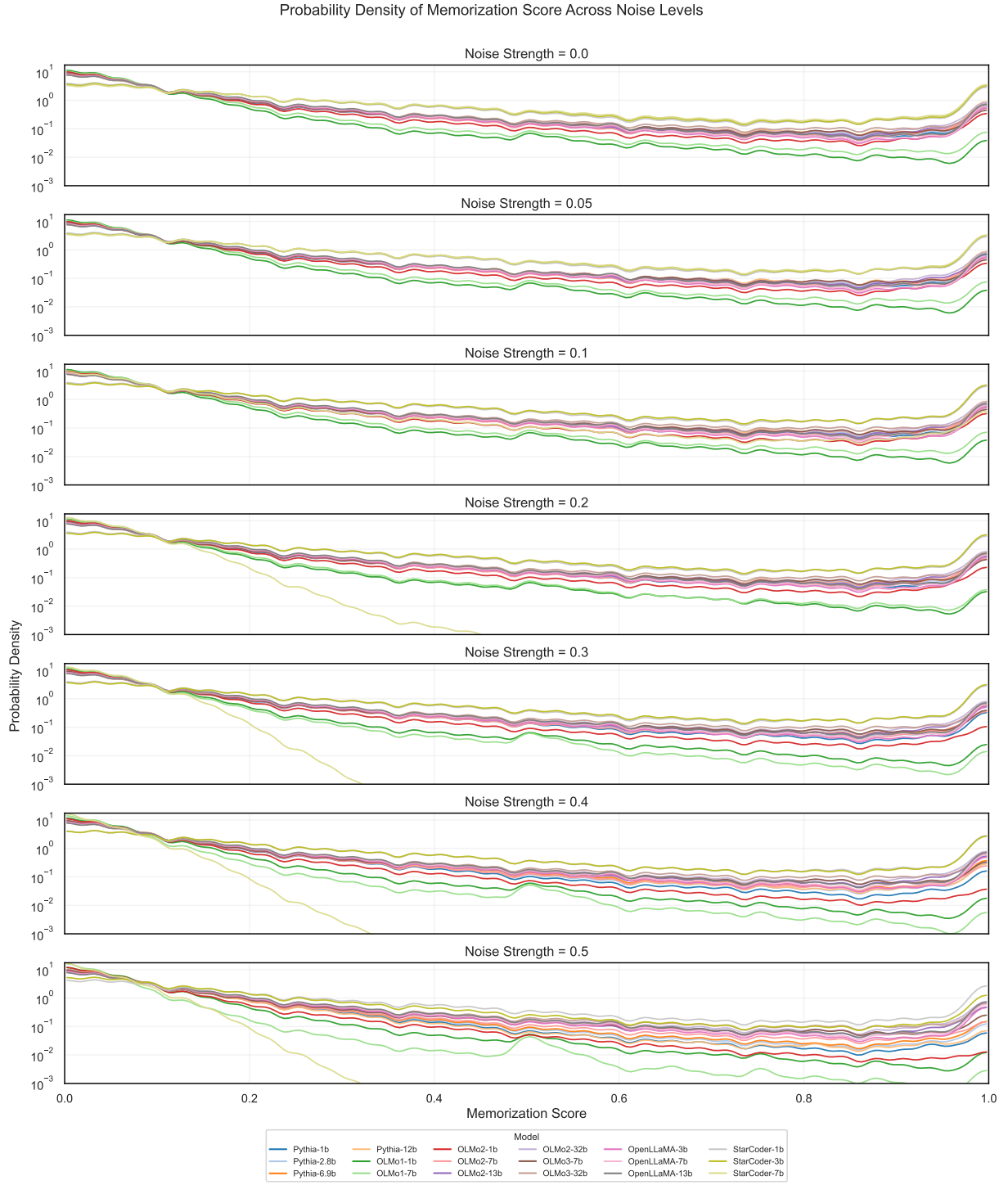


Figure 13: Probability density of the memorization score for all models across noise strengths. Each subplot corresponds to one noise strength (0.0, 0.05, 0.1, 0.2, 0.3, 0.4, and 0.5); the x-axis is the memorization score, and the y-axis is the probability density on a logarithmic scale. Colors denote individual models.

Model	Size	Domain	Count	Share	Total
OLMo1	1b	starcoder	2,834	48.99%	5,785
		cc_en_tail	575	9.94%	
		c4_filtered	367	6.34%	
		books	334	5.77%	
		cc_news_head	318	5.50%	
		proof_pile_2_open_web_math	295	5.10%	
		cc_news_tail	202	3.49%	
		cc_en_middle	185	3.20%	
		proof_pile_2_algebraic_stack	175	3.03%	
		falcon_refinedweb_filtered	118	2.04%	
		cc_news_middle	113	1.95%	
		cc_en_head	97	1.68%	
		wikiref_megawika	53	0.92%	
		wiki	42	0.73%	
		redpajama_arxiv	34	0.59%	
		reddit	27	0.47%	
		pes2o	9	0.16%	
		tulu_flan	4	0.07%	
		redpajama_stackexchange	3	0.05%	
	7b	starcoder	3,952	35.01%	11,287
		cc_en_tail	994	8.81%	
		proof_pile_2_open_web_math	981	8.69%	
		books	940	8.33%	
		c4_filtered	830	7.35%	
		proof_pile_2_algebraic_stack	679	6.02%	
		cc_news_tail	613	5.43%	
		cc_news_head	530	4.70%	
		cc_en_middle	342	3.03%	
		cc_news_middle	310	2.75%	
		falcon_refinedweb_filtered	215	1.90%	
		wiki	207	1.83%	
		cc_en_head	196	1.74%	
		wikiref_megawika	150	1.33%	
		redpajama_arxiv	92	0.82%	
		tulu_flan	92	0.82%	
		reddit	72	0.64%	
		pes2o	66	0.58%	
		redpajama_stackexchange	26	0.23%	
	13b	flan	17,133	24.80%	69,080
		starcoder	14,447	20.91%	
		math	10,152	14.70%	
		algebraic-stack	7,072	10.24%	
		open-web-math	5,439	7.87%	
		arxiv	4,000	5.79%	
		dclm	3,251	4.71%	
		default	2,679	3.88%	
		pre_train_dclm	1,804	2.61%	
		wiki	1,377	1.99%	

Continued on next page

Model	Size	Domain	Count	Share	Total
		pre_train_wiki	1,293	1.87%	39,219
		stackexchange	280	0.41%	
		pre_train_pes2o	81	0.12%	
		pes2o	72	0.10%	
	1b	flan	12,291	31.34%	
		starcoder	8,188	20.88%	
		math	5,223	13.32%	
		algebraic-stack	3,868	9.86%	
		open-web-math	3,766	9.60%	
		arxiv	2,457	6.26%	
		dclm	1,217	3.10%	
		default	1,169	2.98%	
		pre_train_dclm	600	1.53%	
		pre_train_wiki	184	0.47%	
		wiki	159	0.41%	
		stackexchange	83	0.21%	
		pes2o	10	0.03%	
		pre_train_pes2o	4	0.01%	
	32b	flan	19,828	20.59%	
		starcoder	19,185	19.93%	
		math	11,390	11.83%	
		algebraic-stack	10,433	10.84%	
		dclm	8,958	9.30%	
		open-web-math	7,130	7.41%	
		arxiv	5,389	5.60%	
		default	4,527	4.70%	
		pre_train_dclm	3,624	3.76%	
		wiki	2,593	2.69%	
		pre_train_wiki	2,451	2.55%	
		stackexchange	589	0.61%	
		pre_train_pes2o	95	0.10%	
		pes2o	92	0.10%	
	7b	flan	14,803	24.46%	
		starcoder	13,729	22.69%	
		math	8,142	13.46%	
		algebraic-stack	7,515	12.42%	
		open-web-math	5,035	8.32%	
		arxiv	3,603	5.95%	
		default	2,297	3.80%	
		dclm	2,256	3.73%	
		pre_train_dclm	1,436	2.37%	
		pre_train_wiki	774	1.28%	
		wiki	727	1.20%	
		stackexchange	161	0.27%	
		pre_train_pes2o	17	0.03%	
		pes2o	14	0.02%	
	stack_edu_Java		12,600	5.15%	
	stack_edu_PHP		10,714	4.38%	

Continued on next page

Model	Size	Domain	Count	Share	Total
		stack_edu_SQL	10,488	4.29%	
		common_crawl_history_and_geography	9,791	4.00%	
		stack_edu_Go	9,535	3.90%	
		finemath_3plus	8,732	3.57%	
		stack_edu_Rust	8,363	3.42%	
		stack_edu_Ruby	8,066	3.30%	
		stack_edu_Python	7,392	3.02%	
		common_crawl_software	6,892	2.82%	
		common_crawl_software_development	6,652	2.72%	
		common_crawl_travel_and_tourism	6,603	2.70%	
		rpj_proofpile_arxiv	6,478	2.65%	
		common_crawl_transportation	6,412	2.62%	
		common_crawl_religion	6,257	2.56%	
		stack_edu_JavaScript	6,075	2.48%	
		common_crawl_literature	6,071	2.48%	
		stack_edu_Cpp	5,986	2.45%	
		stack_edu_C	5,753	2.35%	
		common_crawl_science_math_and_technology	5,729	2.34%	
		stack_edu_Swift	5,701	2.33%	
		common_crawl_home_and_hobbies	5,689	2.32%	
		common_crawl_games	5,676	2.32%	
		common_crawl_finance_and_business	5,569	2.28%	
		common_crawl_electronics_and_hardware	5,467	2.23%	
		common_crawl_entertainment	5,317	2.17%	
		stack_edu_Shell	5,303	2.17%	
		stack_edu_Markdown	5,258	2.15%	
		common_crawl_industrial	4,311	1.76%	
		common_crawl_crime_and_law	4,099	1.68%	
		common_crawl_education_and_jobs	4,000	1.63%	
		common_crawl_health	3,853	1.57%	
		stack_edu_CSharp	3,747	1.53%	
		stack_edu_TypeScript	3,652	1.49%	
		common_crawl_social_life	3,159	1.29%	
		common_crawl_food_and_dining	3,068	1.25%	
		common_crawl_art_and_design	3,043	1.24%	
		common_crawl_adult_content	2,932	1.20%	
		common_crawl_politics	2,806	1.15%	
		common_crawl_sports_and_fitness	1,865	0.76%	
		dolma1_7_wiki_en	1,802	0.74%	
		olmocr_science_pdfs_education_and_jobs	1,711	0.70%	
		common_crawl_fashion_and_beauty	1,506	0.62%	
		olmocr_science_pdfs_art_and_design	282	0.12%	
		olmocr_science_pdfs_entertainment	267	0.11%	
		olmocr_science_pdfs_sports_and_fitness	13	0.01%	
		olmocr_science_pdfs_adult_content	8	0.00%	
		stack_edu_Java	9,975	6.07%	
		stack_edu_SQL	8,621	5.25%	
		stack_edu_Go	7,712	4.69%	
		stack_edu_PHP	7,434	4.52%	

Continued on next page

Model	Size	Domain	Count	Share	Total
		common_crawl_history_and_geography	6,591	4.01%	
		stack_edu_Rust	6,484	3.95%	
		stack_edu_Ruby	5,625	3.42%	
		stack_edu_Python	5,147	3.13%	
		finemath_3plus	4,764	2.90%	
		common_crawl_software_development	4,675	2.84%	
		stack_edu_C	4,493	2.73%	
		rpj_proofpile_arxiv	4,239	2.58%	
		stack_edu_JavaScript	4,209	2.56%	
		stack_edu_Shell	4,193	2.55%	
		common_crawl_software	4,192	2.55%	
		stack_edu_Swift	4,177	2.54%	
		stack_edu_Cpp	4,112	2.50%	
		common_crawl_travel_and_tourism	4,080	2.48%	
		common_crawl_literature	4,038	2.46%	
		common_crawl_games	3,851	2.34%	
		common_crawl_home_and_hobbies	3,764	2.29%	
		common_crawl_entertainment	3,696	2.25%	
		common_crawl_science_math_and_technology	3,640	2.21%	
		common_crawl_transportation	3,525	2.14%	
		common_crawl_electronics_and_hardware	3,490	2.12%	
		common_crawl_religion	3,343	2.03%	
		common_crawl_finance_and_business	3,243	1.97%	
		stack_edu_Markdown	3,087	1.88%	
		common_crawl_crime_and_law	3,040	1.85%	
		common_crawl_industrial	2,820	1.72%	
		stack_edu_CSharp	2,655	1.62%	
		stack_edu_TypeScript	2,586	1.57%	
		common_crawl_education_and_jobs	2,527	1.54%	
		common_crawl_health	2,065	1.26%	
		common_crawl_politics	1,927	1.17%	
		common_crawl_art_and_design	1,885	1.15%	
		common_crawl_social_life	1,807	1.10%	
		common_crawl_adult_content	1,721	1.05%	
		common_crawl_food_and_dining	1,630	0.99%	
		common_crawl_sports_and_fitness	1,040	0.63%	
		common_crawl_fashion_and_beauty	788	0.48%	
		olmocr_science_pdfs_education_and_jobs	672	0.41%	
		dolma1_7_wiki_en	396	0.24%	
		olmocr_science_pdfs_art_and_design	214	0.13%	
		olmocr_science_pdfs_entertainment	154	0.09%	
		olmocr_science_pdfs_adult_content	6	0.00%	
		olmocr_science_pdfs_sports_and_fitness	4	0.00%	
		github_sample	18,335	58.46%	
		common_crawl_sample	5,687	18.13%	
		wikipedia_sample	3,194	10.18%	
	13b	c4_sample	3,137	10.00%	31,365
		arxiv_sample	936	2.98%	
		stackexchange_sample	76	0.24%	

Continued on next page

Model	Size	Domain	Count	Share	Total
	3b	github_sample	15,029	65.76%	22,855
		common_crawl_sample	2,797	12.24%	
		c4_sample	2,211	9.67%	
		wikipedia_sample	2,121	9.28%	
		arxiv_sample	654	2.86%	
		stackexchange_sample	43	0.19%	
	7b	github_sample	16,290	61.05%	26,681
		common_crawl_sample	3,982	14.92%	
		wikipedia_sample	2,877	10.78%	
		c4_sample	2,749	10.30%	
		arxiv_sample	726	2.72%	
		stackexchange_sample	57	0.21%	
	12b	Github	8,380	78.71%	10,647
		Pile-CC	1,427	13.40%	
		FreeLaw	420	3.94%	
		USPTO Backgrounds	134	1.26%	
		PubMed Central	127	1.19%	
		NIH ExPorter	47	0.44%	
		Wikipedia (en)	40	0.38%	
		Enron Emails	27	0.25%	
		StackExchange	27	0.25%	
		ArXiv	6	0.06%	
		HackerNews	4	0.04%	
		Gutenberg (PG-19)	3	0.03%	
		EuroParl	2	0.02%	
		PhilPapers	1	0.01%	
		PubMed Abstracts	1	0.01%	
		Ubuntu IRC	1	0.01%	
	160M	Github	2,082	91.40%	2,278
		FreeLaw	107	4.70%	
		Pile-CC	51	2.24%	
		PubMed Central	36	1.58%	
		PhilPapers	1	0.04%	
		StackExchange	1	0.04%	
	1b	Github	6,929	82.80%	8,368
		Pile-CC	815	9.74%	
		FreeLaw	353	4.22%	
		PubMed Central	125	1.49%	
		USPTO Backgrounds	60	0.72%	
		NIH ExPorter	45	0.54%	
		StackExchange	14	0.17%	
		Wikipedia (en)	10	0.12%	
		Enron Emails	7	0.08%	
		ArXiv	3	0.04%	
		Gutenberg (PG-19)	3	0.04%	
		HackerNews	3	0.04%	
		PhilPapers	1	0.01%	

Continued on next page

Model	Size	Domain	Count	Share	Total
	2.8b	Github	7,575	82.81%	9,147
		Pile-CC	805	8.80%	
		FreeLaw	369	4.03%	
		USPTO Backgrounds	141	1.54%	
		PubMed Central	126	1.38%	
		NIH ExPorter	45	0.49%	
		Wikipedia (en)	29	0.32%	
		Enron Emails	26	0.28%	
		StackExchange	18	0.20%	
		ArXiv	5	0.05%	
		HackerNews	4	0.04%	
		PubMed Abstracts	2	0.02%	
		Gutenberg (PG-19)	1	0.01%	
		PhilPapers	1	0.01%	
	410M	Github	4,995	85.24%	5,860
		Pile-CC	411	7.01%	
		FreeLaw	254	4.33%	
		PubMed Central	116	1.98%	
		NIH ExPorter	45	0.77%	
		USPTO Backgrounds	25	0.43%	
		StackExchange	9	0.15%	
		ArXiv	1	0.02%	
		Enron Emails	1	0.02%	
		Gutenberg (PG-19)	1	0.02%	
		PhilPapers	1	0.02%	
		Wikipedia (en)	1	0.02%	
	6.9b	Github	8,350	79.29%	10,531
		Pile-CC	1,376	13.07%	
		FreeLaw	405	3.85%	
		PubMed Central	128	1.22%	
		USPTO Backgrounds	127	1.21%	
		NIH ExPorter	46	0.44%	
		Enron Emails	27	0.26%	
		Wikipedia (en)	27	0.26%	
		StackExchange	23	0.22%	
		ArXiv	7	0.07%	
		HackerNews	7	0.07%	
		Gutenberg (PG-19)	4	0.04%	
		PubMed Abstracts	2	0.02%	
		PhilPapers	1	0.01%	
		Ubuntu IRC	1	0.01%	
	1b	scala	69,591	16.30%	426,831
		java	59,360	13.91%	
		html	43,000	10.07%	
		c	42,895	10.05%	
		go	38,165	8.94%	
		kotlin	28,411	6.66%	
		c_sharp	18,946	4.44%	

Continued on next page

Model	Size	Domain	Count	Share	Total
		php	18,309	4.29%	
		rust	17,913	4.20%	
		python	16,425	3.85%	
		javascript	11,033	2.58%	
		swift	9,282	2.17%	
		shell	8,943	2.10%	
		typescript	8,793	2.06%	
		sql	7,888	1.85%	
		ruby	7,659	1.79%	
		lua	7,534	1.77%	
		css	5,822	1.36%	
		dockerfile	3,779	0.89%	
		julia	2,811	0.66%	
		r	272	0.06%	
		scala	72,282	15.24%	
		java	63,976	13.49%	
		c	48,425	10.21%	
		html	45,264	9.55%	
		go	41,146	8.68%	
		kotlin	31,384	6.62%	
		c_sharp	22,748	4.80%	
		php	21,139	4.46%	
		rust	20,112	4.24%	
		python	18,807	3.97%	
3b		javascript	12,895	2.72%	474,193
		lua	10,242	2.16%	
		shell	10,195	2.15%	
		sql	10,172	2.15%	
		swift	10,031	2.12%	
		typescript	9,652	2.04%	
		ruby	9,273	1.96%	
		css	7,262	1.53%	
		dockerfile	4,688	0.99%	
		julia	3,789	0.80%	
		r	711	0.15%	
		scala	74,377	14.17%	
		java	68,861	13.12%	
		c	53,398	10.18%	
		html	49,576	9.45%	
		go	44,513	8.48%	
		kotlin	34,436	6.56%	
		c_sharp	25,344	4.83%	
		php	24,442	4.66%	
		rust	22,102	4.21%	
		python	20,861	3.98%	
7b		javascript	15,167	2.89%	524,708
		lua	13,309	2.54%	
		sql	12,755	2.43%	

Continued on next page

Model	Size	Domain	Count	Share	Total
		shell	11,726	2.23%	
		typescript	11,561	2.20%	
		swift	11,075	2.11%	
		ruby	10,645	2.03%	
		css	9,104	1.74%	
		dockerfile	5,755	1.10%	
		julia	4,849	0.92%	
		r	852	0.16%	

Table 7: Domain distribution of memorized texts across model families and sizes.