LLMs Can Leak Training Data But Do They Want To? A Propensity-Aware Evaluation of Memorization in LLMs

Anonymous ACL submission

Abstract

Large language models can reproduce training data, but existing memorization evaluations mostly measure whether models can be forced to do so, rather than whether they do so under ordinary use. We introduce PROPME, a propensity-aware framework for memorization evaluation that contrasts prefix-based capability attacks with non-adversarial evaluations. We propose a metric transformation that, applied to existing functions, allows to create propensity metrics. We further introduce SIMPLETRACE, a lightweight tracing pipeline built on infini-gram that deterministically attributes model generations to large-scale training corpora and computes verbatim, near-verbatim, and propensity-transformed memorization metrics. Evaluating two fully-open models: Comma and DFM Decoder on two datasets: Common Pile and Dynaword in two languages, we find a consistent gap between capability and propensity: prefix attacks elicit substantially stronger memorization signals than generic or dataset-specific prompts, while propensity scores remain low overall. Thus, the models can reveal training data when directly elicited, but rarely do so in more common non-adversarial settings. We also find that DFM Decoder, which is continually pre-trained from Comma, exhibits reduced memorization and memorization propensity for Common Pile, confirming that memorization capability can decrease when later training emphasizes partially different data. Our results suggest, and we encourage, that memorization audits should report both worst-case extractability and ordinary leakage propensity in order to have a more comprehensive view of this phenomenon.

1 Introduction

Memorization in large language models (LLMs) has been extensively documented (Carlini et al., 2021, 2023): models have been shown to regenerate copyrighted books (Ahmed et al., 2026; Karamolegkou et al., 2023) and sensitive personal identifiers (Carlini et al., 2021), making a thorough understanding of this behaviour critical for safe and ethical deployment. Existing work approaches memorization through adversarial attacks: membership inference (Shokri et al., 2017), prefix attacks (Kiyomaru et al., 2024; Cooper et al., 2025; Ahmed et al., 2026), resource-referencing prompts (Karamolegkou et al., 2023), and divergence attacks (Nasr et al., 2025), and through analysis of factors that modulate it, including data duplication (Kandpal et al., 2022), training time (Huang et al., 2024), and fine-tuning (Kassem et al., 2025). Previous work shows that models *can* reproduce training data under elicitation, i.e., it characterises memorization as a *capability*. Far less attention has been paid to memorization *propensity*: whether models *will* reproduce training data in ordinary, non-adversarial use. Aerni et al. (2024) take a step in this direction by testing memorization with non-adversarial prompts, but their analysis is restricted to closed models, relies on web snippets rather than direct comparison against training data and does not compares ordinary with adversarial settings, limiting both accuracy and analysis scope. We address these gaps by providing a full pipeline that traces model outputs back to the original training corpus and by introducing evaluation settings that span the full spectrum from propensity-focused (generic, non-adversarial prompts) to capability-focused (prefix attacks), enabling a principled comparison of the two.

Evaluating memorization under non-adversarial settings is useful not only to better understand model behaviour but also to support legal compliance. For example, in the context of the European Union, the GDPR (General Data Protection Regulation) (European Parliament and Council of the European Union, 2016, Arts. 5(1)(f), 5(2), 25, 32) requires integrity, confidentiality, accountability, data protection by design, and regular testing of

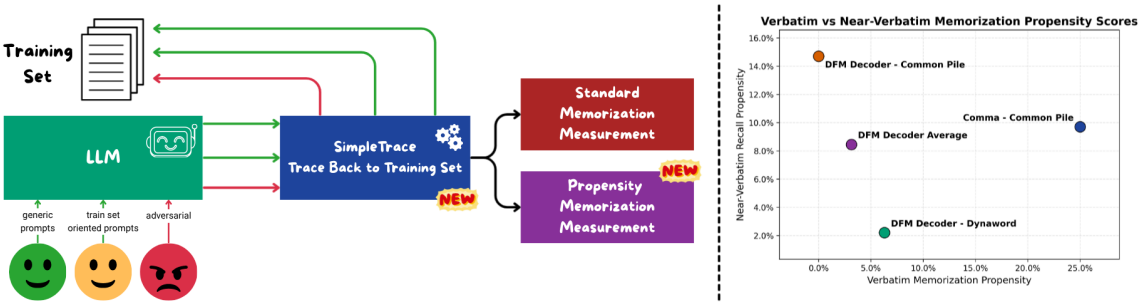


Figure 1: **Left:** PROPME framework overview with propensity and capability prompts, back-tracing to full training set and memorization/propensity measurements. **Right:** propensity metrics results for different combinations of models and dataset, this tells us what is the propensity of a given model to leak data of a certain dataset. The metrics used are defined and detailed in Sections 2, 3.2 4.3

security measures, while the EU AI Act (European Parliament and Council of the European Union, 2024, Arts. 9, 15, 55) requires risk management, robustness, cybersecurity, and, for certain models, evaluation to identify and mitigate systemic risks. Thus, assessing a model's propensity to reproduce training data under ordinary use can provide evidence that foreseeable leakage risks.

Here, we propose PROPME, a framework for systematic evaluation of memorization in large language models with a focus on capability vs. propensity. PROPME comprises three levels of analysis (multi-level), with prompts ranging from generic inputs, focusing on propensity, to prefix attacks (Karamolegkou et al., 2023), focusing on capabilities. Alongside PROPME, we introduce SIMPLETRACE (Section 3.3), an open-source, lightweight tool inspired by OLMoTrace (Liu et al., 2025) and built on infini-gram (Liu et al., 2024) for fast and parallel tracing of model text outputs against large-scale training data. By enabling direct search over the training corpus, SIMPLETRACE provides deterministic attribution and eliminates the ambiguity of probabilistic detection. Deterministic attribution allows precise identification of which training documents a given text was memorized from.

Contributions Our contributions can be summarized as follows:

- We introduce PROPME, the first framework for propensity-aware evaluation of memorization in large language models (Figure 1), featuring multiple settings going from propensity- to capability-focused evaluation. This allows to evaluate and compare the willingness of a model to leak data in real-world prompting compared to adversarial attacks.
- We introduce a novel transformation for turning standard evaluation metrics into propensity metrics. We apply this transformation to existing memorization metrics for measuring the propensity of a large language model to leak training data, conditioning on both adversarial and non-adversarial settings.
- We introduce SIMPLETRACE as a foundational tool for tracing model outputs back to large-scale training data, enabling attribution of potentially memorized sequences to their source documents in the training set.
- We provide results demonstrating the usefulness of PROPME and SIMPLETRACE in a multi-lingual and multi-model scenario. Targeting two models trained on public, permissibly licensed data from two datasets: Comma as a monolingual English model and DFM Decoder as an exemplary model that is continually trained from Comma on a lower resource language (Danish). This enables us to study the effect of continual pre-training on memorization propensity with respect to the original corpus and the new corpus, respectively.

2 Related Work

Memorization Current research can be categorized along two axes: target model type and measurement method. Target model types range from closed or commercial models (Ahmed et al., 2026) to open models (Carlini et al., 2021; Panda et al., 2025; Cooper et al., 2025). Measurement methods vary from model-internal approaches, such as activations, weights, or output probabilities (Huang et al., 2024; Shi et al., 2024; Zhang et al., 2024;

Menta et al., 2025), to comparisons with external texts, such as books or training data (Kassem et al., 2025; Kandpal et al., 2022; Kiyomaru et al., 2024). More broadly, existing work focuses on *detection*, predicting whether a sequence was seen during training (Shi et al., 2024; Zhang et al., 2024), or *extraction*, recovering training sequences through adversarial or targeted prompting (Carlini et al., 2021; Panda et al., 2025). While related work shows that models *can* reproduce memorized content, it largely evaluates memorization as a *capability*. Less is known about memorization *propensity*: whether models tend to reproduce training data under ordinary or weakly targeted conditions (Romero-Alvarado et al., 2026; Voudouris et al., 2026).

Propensity vs Capability in Large Language Models Recent work has argued that LLM evaluations should distinguish between *capabilities*, i.e., behaviours that models can exhibit when successfully elicited, and *propensities*, i.e., behaviours that models tend to exhibit under a given distribution of contexts (Romero-Alvarado et al., 2026; Voudouris et al., 2026). Most existing evaluations are capability-focused: they measure upper bounds on model behaviour through benchmarks, adversarial prompting, red-teaming, or elicitation procedures (Shevlane et al., 2023; Greenblatt et al., 2024; Hofstätter et al., 2025). However, capability-focused evaluations may not predict deployment behaviour, since models can hide or fail to reveal latent capabilities (Greenblatt et al., 2024; Hofstätter et al., 2025), strategically underperform (van der Weij et al., 2024), or adapt their behaviour when they detect evaluation settings (Needham et al., 2025). This gap has motivated propensity-aware evaluations, particularly in agentic safety, where studies distinguish whether models are merely capable of scheming or misalignment from whether they are likely to exhibit such behaviours under realistic prompts, goals, tools, and oversight conditions (Meinke et al., 2024; Hopman et al., 2026; Naik et al., 2025; Järviniemi et al., 2026). Our work adopts this distinction for the first time in memorization, evaluating not only whether models can reproduce training data under elicitation, but also whether they tend to do so in non-adversarial settings.

Memorization Metrics Based on Text/Token Comparison Several metrics have been proposed to quantify memorization via text- or token-level comparison. **Verbatim memorization length** (Huang et al., 2024) measures the maximum number of tokens in the model's greedy continuation that exactly match the target, declaring a sequence memorized when at least 32 tokens are reproduced verbatim from a prefix of at most 32 tokens. **Fraction of extractable sequences** (Carlini et al., 2023) reports the fraction of suffixes reproduced verbatim when the model is conditioned on the corresponding prefix. **LCS** (Karamolegkou et al., 2023) measures the longest common subsequence between the generation and the gold text. **Near-verbatim recall (nv-recall)** (Ahmed et al., 2026) identifies sufficiently long near-verbatim matching blocks between a generation G and a reference B, merges nearby blocks, filters short matches, and computes $\text{nv-recall}(B, G) = m/|B|$, where m is the total word count of retained in-order matches. Some more metrics we considered but are less relevant for this work are in Appendix F.

Tracing Training Set Data Infini-gram (Liu et al., 2024) utilizes a modernized n-gram language model powered by suffix arrays to scale to trillions of tokens, enabling millisecond-latency n-gram counting and probability estimation over arbitrarily long contexts. Building on this infrastructure, OLMoTrace (Liu et al., 2025) provides a real-time system for tracing large language model's generations back to their large training corpora. By detecting and highlighting verbatim matches between model-generated segments and source training documents, it supports the evaluation of model behaviors such as fact-checking, hallucination, and creativity through direct grounding in training data.

Relationship to OLMoTrace and Prior Scripts SIMPLETRACE is directly inspired by OLMoTrace (Liu et al., 2025) but targets offline, systematic large-scale analysis rather than interactive single-input tracing. To the best of our knowledge, the OLMoTrace codebase is not publicly available; only a thin infini-gram API wrapper has been released.[1] Relative to prior informal tracing scripts (Wolfe, 2025), SIMPLETRACE adds an indexing step to build the suffix-array index over the training corpus and a unigram precomputation step required for rarity-based span filtering. Relative to both prior scripts and OLMoTrace it adds a multi-worker parallelization for batch processing, and a metrics calculation and aggregation step that produces in-

[1] https://github.com/allenai/infinigram-api

terpretable summary statistics. SIMPLETRACE is released as open-source.

3 Proposed Method: Propensity-Aware Memorization Evaluation

3.1 Propensity-Capability Evaluation Settings

Enabling propensity-aware evaluation of a behavior b in a model M requires observing the elicitation of b across a range of conditions. Specifically, it is necessary to consider both settings in which the model operates under ordinary, realistic conditions and settings designed to maximally elicit b through targeted interventions. Only by contrasting these two extremes can one obtain a comprehensive and unbiased characterization of M's behavior across the full spectrum from propensity to capability.

In the context of memorization evaluation, we propose assessing M under two prompting scenarios. The first consists of plausible, real-world prompts that are not drawn from the training set and exhibit low lexical overlap with it, targeting the model's propensity to reproduce training data under ordinary use. We call it a *propensity setting*. The second follows the prefix-attack paradigm (Karamolegkou et al., 2023), where the model is conditioned on prefixes of sequences extracted directly from the training set, targeting the model's capability to reproduce memorized content under adversarial elicitation: a *capability setting*.

3.2 Propensity Metrics

We argue that a complete measure of a model's propensity toward a given behavior must also account for its capability to exhibit that behavior. The intuition is as follows. Let b denote a behavior of interest in a model M, and let $f_b \in [0,1]$ be a scalar metric quantifying the extent to which b is exhibited. We consider two evaluation settings: a propensity setting p, where the model is prompted under realistic, non-adversarial conditions, and a capability setting c, where the model is prompted to maximally elicit behavior b (e.g., via a prefix attack designed to induce training data leakage). Let $f_b^p(M,x)$ and $f_b^c(M,x)$ denote the values of f_b observed in settings p and c respectively, for a given input x.

We argue that, given a fixed low value of $f_b^p(M,x)$, observing a high value of $f_b^c(M,x)$ is evidence of *lower* propensity than observing a low one. That is, when a model demonstrates high capability for behavior b under adversarial elicitation, yet does not exhibit b under ordinary prompting, the latter is reinforced as a meaningful signal that the model is not inherently inclined toward b. To operationalize this reasoning, we introduce a *propensity-aware* transformation that, given a behavior b, a propensity setting p, a capability setting c, and a base metric $f_b \in [0,1]$, produces a propensity metric $PM_{f_b} \in [0,1]$[2]:

$$PM_{f_b}(M,x) = \frac{1}{2} \cdot \left(1 + \frac{f_b^p(M,x) - f_b^c(M,x)}{f_b^p(M,x) + f_b^c(M,x)}\right) \quad (1)$$

with $PM_{f_b}(M,x) = 0$ when $f_b^p(M,x) = 0$. An interpretation of the metric for a model M is:

- **High capability, low propensity** (f_b^c high, f_b^p low): PM_{f_b} is low. Although the model is capable of exhibiting b under elicitation, the behavior is largely absent under ordinary prompting, indicating low propensity.

- **Low capability, high propensity** (f_b^c low, f_b^p high): PM_{f_b} is high. Even though b is not strongly elicited in the capability setting, it manifests spontaneously under propensity conditions, indicating a strong propensity.

- **Equal values in both settings**: $PM_{f_b} = 0.5$, a neutral score. The model shows consistent behavior across settings, with propensity neither amplified by low capability nor attenuated by high capability. Having values at 0 in the propensity setting always gives $PM_{f_b} = 0$ as no propensity is manifested.

We apply this transformation in Section 4 to existing memorization metrics, turning them into propensity memorization metrics.

Propensity degree $\neq$ behavior degree. Note that this metric is aimed at capturing the degree of *propensity* to manifest b and not the degree of *manifesting* b itself (e.g. memorization/leakage). Hence, a high value for a propensity metric for b suggests just high tendency of b (under standard settings) and not that the model is e.g. always manifesting b. Taking the case of memorization, while the **manifestation** degree is already captured by standard metrics, the **propensity** degree has not yet been well defined.

[2] https://www.desmos.com/calculator/zrbjlk0s2u

3.3 SIMPLETRACE: Enabling Accurate Memorization Evaluation

SIMPLETRACE is built on top of the infini-gram engine (Liu et al., 2024) for fast n-gram queries over suffix-array indexes of large corpora, and follows the OLMoTrace pipeline (Liu et al., 2025); differences are discussed in Section 2. The pipeline consists of four steps, augmented with multi-worker parallelization and a metrics aggregation step.

Step 1 (maximal span extraction) iterates over all $L-1$ suffixes of a generation of L tokens, querying each against the suffix array to recover the longest verbatim prefix appearing in the corpus; candidates are filtered to well-formed, maximal, word-boundary-respecting spans, with a mixed mode for code and math.

Step 2 (unigram rarity filtering) scores each maximal span by the joint unigram probability of its tokens and retains the $K = \lceil 0.05 \cdot L \rceil$ rarest spans, reducing noise from boilerplate matches.

Step 3 (document retrieval) issues a second index lookup for each retained span to retrieve matching training documents, classifying each match as a full raw match, a full normalized match, or a partial span-level match; retrieval is capped via deterministic subsampling.

Step 4 (span merging and aggregation) collapses adjacent or overlapping retained spans into non-redundant segments by a sequential greedy merge, producing the final set of traced regions.

Metrics calculation. SIMPLETRACE computes a comprehensive set of statistics over the full batch of processed generations, producing over 30 summary fields covering span lengths, document retrieval counts, match tiers, and memorization rates (Appendix H). These include average and maximum longest span length, proportions of generation matched verbatim in the training set, span-length distributions, and k-eidetic memorization rates. SIMPLETRACE also implements an *adaptive nv-recall* variant (Ahmed et al., 2026) that scales merge and filter thresholds proportionally to the reference document length, ensuring consistent behaviour across the diverse document lengths found in real training corpora without manual tuning.

Validation We validate SIMPLETRACE with unit tests on a dummy corpus and with end-to-end experiments on Common Pile (Kandpal et al., 2026) and Dynaword (Enevoldsen et al., 2025). For each sampled document, we evaluate one full-document query and three partial queries anchored at the start, middle, and end. For each, we measure both source-document retrieval and exact text matching. SIMPLETRACE achieves perfect retrieval and exact-match results on Dynaword, and near-perfect source-document retrieval with exact span recovery for all Common Pile queries, including perfect full-document recovery. Full results are available in Appendix A. Running 100 queries with SIMPLETRACE takes approx. 1 minute for Common Pile (large approx. 460B tokens) using 4 CPU cores, and approx. 10 seconds for Dynaword (large approx. 6.8B tokens) using 10 CPU cores.

4 Experimental Setup

All experiments are conducted with temperature 0, using greedy decoding throughout.

4.1 Datasets, Models, and Indexes

We index two datasets using infini-gram (Liu et al., 2024). **Common Pile** (Kandpal et al., 2026) is represented by the `Comma v0.1` training corpus (521 GB, 463.6B `Comma v0.1` tokens), indexed across three balanced shards using 128 CPU cores and a 350 GB memory budget (requiring approx. 2.5–3 hours per shard). **Danish Dynaword** (Enevoldsen et al., 2025) contains 5.66M samples (6.83B Llama 3 tokens, 10.5 GB) and was indexed using 16 CPU cores and an 84 GB memory budget (≈3 hours). Both datasets consist exclusively of open, permissibly licensed data. We evaluate two models trained on these corpora. **Comma v0.1** (Kandpal et al., 2026) is pre-trained on the Comma dataset. **DFM Decoder Open v0**[3] is a continual pre-training of `Comma v0.1` over 30B tokens in three stages, with a fixed data mixture of two-thirds Dynaword and one-third Common Pile throughout. This pair allows us to study memorization along two axes: language (English vs. Danish) and training stage. Stage 1 used a batch size of 262 144 tokens for 37 852 steps (lr $=$ 1e$-$5, constant); Stages 2 and 3 doubled the batch size to 524 288 tokens over 18 926 steps each, with Stage 3 applying square-root decay from 1e$-$5. The released checkpoint corresponds to Stage 3.

4.2 Propensity and Capability Settings

We define three evaluation settings for each dataset, each corresponding to a distinct prompt

[3] https://huggingface.co/danish-foundation-models/dfm-decoder-open-v0-7b-pt

set: `Generic`, `Specific`, and `Prefix`, all containing 100 samples. The first two are designed to elicit memorization propensity: they consist of plausible, naturally-phrased prompts with low expected overlap with the training data. The third targets memorization capability, following the prefix-attack setting of Karamolegkou et al. (2023): prompts are constructed by extracting random training examples of at least 100 tokens and conditioning the model on their first 50 tokens; generations are then evaluated against the full training set.

The `Generic` and `Specific` prompt sets were generated using GPT-5.5 (OpenAI, 2026). For both sets, the model was instructed to produce plausible prompts given the domain of the respective training dataset. For `Specific`, the URL of the dataset was additionally provided, and the model was explicitly instructed to generate prompts inspired by but not extracted from the dataset. Full prompting instructions are reported in Appendix E.

We validate all three prompt settings using SIMPLETRACE (Section 3.3) to quantify their overlap with the training data prior to any memorization evaluation. The automatically generated prompt sets exhibit substantially lower training-data overlap than the `Prefix` set, while the `Specific` prompts display higher overlap than `Generic` ones (Appendix B). Both non-adversarial sets therefore constitute suitable conditions for measuring the propensity of models to reproduce training data under realistic, non-targeted prompting.

4.3 Metrics

We evaluate memorization using four metrics computed by SIMPLETRACE. *Average longest span length* (Karamolegkou et al., 2023) is the mean, over all generations, of the longest verbatim span found in each generation. *Generations full matches ratio* (Carlini et al., 2023) is the fraction of generations for which at least one retrieved training document contains the full generation verbatim. *Average nv-recall* (Ahmed et al., 2026) is the mean adaptive nv-recall (Section 3.3) across all retrieved documents. Together these metrics cover both verbatim and near-verbatim reproduction, as advocated by Huang et al. (2024) and Ippolito et al. (2023). We further apply the propensity-aware transformation from Section 3.2 to *generations full matches ratio* and *average nv-recall*, yielding propensity-aware variants that jointly characterise memorization behaviour under both ordinary and adversarial prompting conditions. Average near-verbatim recall will also be referenced as `NVR` or nv-recall; average longest span as `ALS` and generations full matches ratio as `FMR` or full matches ratio.

Prompt	NVR	FMR	ALS
Generic	0.0013	0.0000	27.95
Specific	0.0058	**0.0200**	29.41
Prefix	**0.0321**	**0.0200**	**50.35**

Table 1: Memorization Analysis for Common Pile on Comma across prompt settings.

Propensity Metric	Generic	Specific
PM_{NVR}	0.0402	**0.1528**
PM_{FMR}	0	**0.5000**

Table 2: Propensity memorization scores for Common Pile on Comma model.

5 Results

5.1 Memorization of Pre-Training Data

Table 1 shows core SIMPLETRACE metrics for Common Pile memorization in the Comma model. Table 2 shows the corresponding propensity scores.

Non-adversarial memorization is non-negligible but dominated by prefix attacks. Prefix attacks yield the strongest memorization signal, with `ALS` of 50.35 tokens versus 27.95 (generic) and 29.47 (specific). However, the non-adversarial settings are not negligible relative to prefix attacks: `NVR` reaches 0.032 (prefix), 0.006 (specific), and 0.001 (generic). Notably, specific prompts match the prefix attack `FMR` of 0.02, suggesting that even weakly targeted prompts can occasionally be as effective as prefix attacks at eliciting complete verbatim reproductions from this corpus.

Comma has non-negligible propensity of reproducing training data under specific setting. For `NVR`, generic and specific propensity scores are 0.040 and 0.153 respectively, both well below 0.5, yet noticeably higher than corresponding scores of DFM Decoder on Dynaword (Section 4), reflecting the larger non-adversarial signal in this experiment. The `FMR` generic propensity is 0; the specific setting yields 0.5. Accordingly to our metric definition (Section 3.2), this is due to identical full-match rates in specific and prefix settings, which, however are are both relatively low (0.02).

That is, even if verbatim generation of training data is low, the model has the propensity of generating it under prompting settings that, even if

Model	Dataset	Prompt	NVR	FMR	ALS
Comma	CP	Generic	0.0013	0	27.95
		Specific	0.0058	**0.0200**	29.47
		Prefix	**0.0321**	**0.0200**	**50.35**
DFM	CP	Generic	0.0003	0	23.57
		Specific	0.0093	0	30.15
		Prefix	0.0239	0	**40.83**
	DW	Generic	0.0010	0	15.68
		Specific	0.0007	0.0100	17.37
		Prefix	**0.0363**	**0.0700**	**24.75**

Table 3: Memorization metrics for all model–corpus pairs across prompt settings. CP: Common Pile, DW: Dynaword.

Model	Dataset	Metric	Generic	Specific
Comma	CP	PM_{NVR}	0.0402	0.1528
		PM_{FMR}	0	**0.5000**
DFM	CP	PM_{NVR}	0.0134	**0.2807**
		PM_{FMR}	0	0
	DW	PM_{NVR}	0.0263	0.0182
		PM_{FMR}	0	**0.1250**

Table 4: Propensity memorization scores for all model–corpus pairs. Scores are computed relative to the prefix (capability) setting. Lower scores indicate lower propensity relative to capability. CP: Common Pile, DW: Dynaword.

non-adversarial, are similar to the training set.

5.2 Memorization in Continual Pre-Training

Tables 3 and 4 report memorization metrics and propensity-aware scores for all model–corpus combinations across the three prompt settings. We repeat values from previous tables so to improve readability and facilitate comparison.

Memorization is substantially higher under prefix attacks. Prefix attacks elicit markedly stronger memorization signals than either non-adversarial setting across all models and corpora. For DFM Decoder on Dynaword, NVR reaches 0.036 under prefix prompting versus 0.001 for both generic and specific prompts – a 36× difference – and FMR rises to 0.07, compared to 0.00 and 0.01 respectively. Under non-adversarial prompting, all model–corpus pairs show negligible memorization across metrics, confirming that models are capable of reproducing training data under elicitation but do so at a negligible rate in ordinary use.

Common Pile and Dynaword exhibit complementary memorization profiles. For DFM Decoder, Common Pile consistently yields longer average verbatim spans (23.57–40.83 tokens) than Dynaword (15.68–24.75 tokens), across all prompt settings. Dynaword, by contrast, exhibits stronger full-generation memorization under prefix attacks: FMR rises to 0.07 while Common Pile remains at 0 across all settings. This suggests two distinct memorization profiles: Common Pile memorization manifests as longer localized verbatim fragments, while Dynaword memorization produces shorter but occasionally complete generation-level reproductions.

Continual pre-training on Dynaword reduces Common Pile memorization. Comma produces longer verbatim spans than DFM Decoder on Common Pile under both generic prompts (27.95 vs. 23.57 tokens) and prefix attacks (50.35 vs. 40.83 tokens), and is the only model to exhibit, on Common Pile, non-zero full-generation memorization (FMR $= 0.02$ under specific prompts and prefix attacks), while DFM Decoder remains at 0 throughout. This is consistent with Kiyomaru et al. (2024), who show that memorization is less likely for texts not encountered in the latter stages of training: as DFM Decoder is continually pre-trained from Comma with a data mixture two-thirds Dynaword and one-third Common Pile, it progressively loses memorization capability on Common Pile. Interestingly, we observe in Table 4 that a strong drop in verbatim propensity (-0.5) corresponded with a weaker increase in near-verbatim propensity ($+0.1279$), indicating progressive shift from a stronger (verbatim) memorization to a weaker one (near-verbatim). A similar pattern can be seen also in standard memorization metrics (Table 3) but is more difficult to notice than in propensity ones. These results further suggest that a balanced multi-dataset training mixture may mitigate memorization across all constituent corpora.

Propensity-aware evaluation reveals a universally low tendency to leak data. For DFM Decoder propensity scores are substantially below the neutral value of 0.5 across all training sets and non-adversarial settings (Table 4). For DFM Decoder on Dynaword, $PM_{\mathrm{NVR}} = 0.026$ (generic) and 0.018 (specific); PM_{FMR} reaches at most 0.125 under specific prompts. For Common Pile, DFM Decoder achieves a specific PM_{NVR} of 0.281, reflecting that targeted-but-non-adversarial prompts recover a non-negligible fraction of the prefix-elicited near-verbatim signal, yet this remains well below neutral. These results confirm that DFM De-

coder does not have a strong tendency to reproduce training data under ordinary prompting conditions, despite demonstrable capability to do so under adversarial elicitation.

5.3 Memorization throughout Training

We evaluate memorization of both Dynaword and Common Pile separately across the three training stages of DFM Decoder (Stage 1, Stage 2, and the final checkpoint).

Across both corpora, memorization profiles are essentially unchanged from Stage 1 through the final checkpoint. For Dynaword, ALS is identical at 15.68, 17.37, and 24.75 tokens for generic, specific, and prefix settings respectively; NVR and FMR vary only minimally with no directional trend; propensity scores are similarly flat (generic $PM_{\text{nv-recall}}$ ranging between 0.023 and 0.027 across stages). Results for Common Pile are analogous: ALS values are stable at 23.57, 30.15, and 40.83 tokens per setting, NVR varies by less than 0.001 across stages in every prompt setting, and FMR is 0 throughout. Since across stages the same data mix is used, this stability is consistent with prior evidence that memorization is tied to when examples are last encountered during training (Kiyomaru et al., 2024), as evidenced also by the memorization signal decrease on Common Pile after continual pretraining of DFM Decoder (Section 5.2). Notably, these results suggest that one training stage is enough for having an impact on memorization of data from previous trainings. Full results are in Appendix C.

6 Discussion

Our results show a clear separation between memorization capability and memorization propensity. Memorization is substantially stronger in capability settings: prefix attacks consistently elicit higher near-verbatim recall, more full-generation matches, and longer verbatim spans than generic or specific prompts. This indicates that the evaluated models can reproduce training data when directly conditioned on it, but that this behavior is much less likely to appear under ordinary prompting.

Propensity is overall low across datasets and models. In common non-adversarial settings, the models rarely reveal memorized data, suggesting that memorization capability alone overstates practical leakage risk. At the same time, low propensity does not imply absence of memorization: specific prompts can still recover memorized content in some cases, and therefore propensity evaluation should complement, not replace, adversarial extraction tests.

The comparison between Comma and DFM Decoder further confirms that accessible memorization can decrease after training on partially different data. DFM Decoder shows weaker memorization of Common Pile than its parent model, while memorization remains comparatively stable across later DFM training stages. This confirms that changes in the training mixture can reduce the accessibility of previously memorized content, and suggests that continued training on the same mixture does not necessarily increase memorization monotonically.

7 Conclusion

We introduced PROPME, a framework for measuring memorization propensity by comparing ordinary prompting settings with adversarial capability settings. Together with SIMPLETRACE, our data attribution pipeline, PROPME enables memorization analysis across verbatim, near-verbatim, and full-generation matches against large-scale training data.

Our experiments show that memorization is much stronger under prefix-based capability evaluations than under non-adversarial propensity evaluations. The models can reveal training data when prompted adversarially, but they rarely do so in more common prompting conditions. We also find that training on a partially different corpus can reduce accessible memorization of earlier data, confirming previous similar work by Kiyomaru et al. (2024). Overall, these findings suggest that memorization audits should report both capability and propensity, since worst-case extractability and ordinary leakage risk capture different aspects of model behavior. Hence, relying only on one aspect does not fully mirror the real model behavior.

8 Limitations

Our focus on direct comparison against full training corpora yields high measurement accuracy but limits applicability to models whose training data is not publicly available. The propensity transformation and the PROPME framework are nevertheless architecture-agnostic and can be combined with logit-, weight-, or probability-based memorization methods when training data access is unavailable. Our experiments cover a single model

family – four checkpoints derived from two base models, three of which are continual pre-trainings of the fourth – and two languages. Extending the analysis to broader model architectures and additional languages would help clarify how architectural choices and multilingual training interact with memorization propensity. Finally, our results leave open the question of how data mixture composition affects memorization: it remains unclear whether mixing same-language data produces effects comparable to those observed here under cross-lingual mixtures of Dynaword and Common Pile.

9 Ethical Considerations

All experiments are conducted on models trained exclusively on open, permissibly licensed data and intended for research use, as in our case. Our findings confirm that adversarial elicitation can surface memorized content even when propensity under ordinary prompting is low, underscoring the importance of capability-level evaluation alongside propensity-level assessment. We release SIMPLE-TRACE as open-source to support transparent and reproducible research. While any tool enabling output-to-training-data tracing could in principle be misused, we believe the accountability benefits outweigh this risk, particularly given that the tool requires full access to the training corpus. Lastly, we argued that memorization propensities (in contrast to capabilities) are important to evaluate and better understand. However, lower memorization propensities should not be used to "green-wash" potential copyright infringement problems. On a longer time horizon, we envision that understanding memorization propensities could be one of several factors for informing copyright law in the future.

References

Michael Aerni, Javier Rando, Edoardo Debenedetti, Nicholas Carlini, Daphne Ippolito, and Florian Tramèr. 2024. Measuring non-adversarial reproduction of training data in large language models. *arXiv preprint arXiv:2411.10242*.

Ahmed Ahmed, A. Feder Cooper, Sanmi Koyejo, and Percy Liang. 2026. Extracting books from production language models. *arXiv preprint arXiv:2601.02671*.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Colin Raffel, and Dawn Song. 2021. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

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*.

Kenneth Enevoldsen, Kristian Nørgaard Jensen, Jan Kostkan, Balázs Szabó, Márton Kardos, Kirten Vad, Andrea Blasi Núñez, Gianluca Barmina, Jacob Nielsen, Rasmus Larsen, and 1 others. 2025. Dynaword: From one-shot to continuously developed datasets. *arXiv preprint arXiv:2508.02271*.

European Parliament and Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).

European Parliament and Council of the European Union. 2024. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending certain Union legislative acts (Artificial Intelligence Act).

Ryan Greenblatt, Fabien Roger, Dmitrii Krasheninnikov, and David Krueger. 2024. Stress-testing capability elicitation with password-locked models. In *Advances in Neural Information Processing Systems 37*.

Felix Hofstätter, Teun van der Weij, Jayden Teoh, Rada Djoneva, Henning Bartsch, and Francis Rhys Ward. 2025. The elicitation game: Evaluating capability elicitation techniques. In *Proceedings of the 42nd International Conference on Machine Learning*.

Mia Hopman, Jannes Elstner, Maria Avramidou, Amritanshu Prasad, and David Lindner. 2026. Evaluating and understanding scheming propensity in llm agents. *arXiv preprint arXiv:2603.01608*.

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.

Daphne Ippolito, Florian Tramer, 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.

Olli Järviniemi, Oliver Makins, Jacob Merizian, Robert Kirk, and Ben Millwood. 2026. Propensity inference: Environmental contributors to llm behaviour. *arXiv preprint arXiv:2604.21098*.

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A Feder Cooper, Aviya Skowron, and 1 others. 2026. The common pile v0. 1: An 8tb dataset of public domain and openly licensed text. *Advances in Neural Information Processing Systems*, 38.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. *arXiv preprint arXiv:2310.13771*.

Aly M Kassem, Omar Mahmoud, Niloofar Mireshghallah, Hyunwoo Kim, Yulia Tsvetkov, Yejin Choi, Sherif Saad, and Santu Rana. 2025. Alpaca against vicuna: Using llms to uncover memorization of 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 8296–8321.

Hirokazu Kiyomaru, Issa Sugiura, Daisuke Kawahara, and Sadao Kurohashi. 2024. A comprehensive analysis of memorization in large language models. In *Proceedings of the 17th International Natural Language Generation Conference (INLG)*.

Jiacheng Liu, Taylor Blanton, Yanai Elazar, Sewon Min, YenSung Chen, Arnavi Chheda-Kothary, Huy Tran, Byron Bischoff, Eric Marsh, Michael Schmitz, and 1 others. 2025. Olmotrace: Tracing language model outputs back to trillions of training tokens. *arXiv preprint arXiv:2504.07096*.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint arXiv:2401.17377*.

Alexander Meinke, Bronson Schoen, Jérémy Scheurer, Mikita Balesni, Rusheb Shah, and Marius Hobbhahn. 2024. Frontier models are capable of in-context scheming. *arXiv preprint arXiv:2412.04984*.

Tarun Ram Menta, Susmit Agrawal, and Chirag Agarwal. 2025. Analyzing memorization in large language models through the lens of model attribution. 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 10661–10689.

Akshat Naik, Patrick Quinn, Guillermo Bosch, Emma Gouné, Francisco Javier Campos Zabala, Jason Ross Brown, and Edward James Young. 2025. Agentmisalignment: Measuring the propensity for misaligned behaviour in llm-based agents. *arXiv preprint arXiv:2506.04018*.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Florian Tramèr, and Katherine Lee. 2025. Scalable extraction of training data from aligned, production language models. In *The Thirteenth International Conference on Learning Representations*.

Joe Needham, Giles Edkins, Govind Pimpale, Henning Bartsch, and Marius Hobbhahn. 2025. Large language models often know when they are being evaluated. *arXiv preprint arXiv:2505.23836*.

OpenAI. 2026. GPT-5.5 System Card. https://deploymentsafety.openai.com/gpt-5-5/gpt-5-5.pdf.

Ashwinee Panda, Xinyu Tang, Milad Nasr, Christopher A. Choquette-Choo, and Prateek Mittal. 2025. Privacy auditing of large language models. In *International Conference on Learning Representations (ICLR)*.

Daniel Romero-Alvarado, Fernando Martínez-Plumed, Lorenzo Pacchiardi, Hugo Save, Siddhesh Milind Pawar, Behzad Mehrbakhsh, Pablo Antonio Moreno Casares, Ben Slater, Paolo Bova, Peter Romero, Zachary R. Tyler, Jonathan Prunty, Luning Sun, and José Hernández-Orallo. 2026. Capabilities ain't all you need: Measuring propensities in ai. *arXiv preprint arXiv:2602.18182*.

Toby Shevlane, Sebastian Farquhar, Ben Garfinkel, Mary Phuong, Jess Whittlestone, Jade Leung, Daniel Kokotajlo, Nahema Marchal, Markus Anderljung, Noam Kolt, Lewis Ho, Divya Siddarth, Shahar Avin, Will Hawkins, Been Kim, Iason Gabriel, Vijay Bolina, Jack Clark, Yoshua Bengio, and 2 others. 2023. Model evaluation for extreme risks. *arXiv preprint arXiv:2305.15324*.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. Detecting pretraining data from large language models. In *International Conference on Learning Representations (ICLR)*.

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 (SP)*, pages 3–18.

Teun van der Weij, Felix Hofstätter, Ollie Jaffe, Samuel F. Brown, and Francis Rhys Ward. 2024. Ai sandbagging: Language models can strategically underperform on evaluations. *arXiv preprint arXiv:2406.07358*.

Konstantinos Voudouris, Mirko Thalmann, Alex Kipnis, José Hernández-Orallo, and Eric Schulz. 2026. Measuring what ai systems might do: Towards a measurement science in ai. *arXiv preprint arXiv:2603.00063*.

Cameron R. Wolfe. 2025. olmo_trace.py: Tracing text using the algorithm proposed by olmotrace. https://gist.github.com/wolfecameron/306aa72a0c5095db460e2ccea9b06777. GitHub Gist, last updated June 19, 2025.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024. Pretraining data detection for large language models: A divergence-based calibration method. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Validation of SIMPLETRACE

SIMPLETRACE is validated at two levels. First, we run controlled unit tests on a dummy index with known document identifiers. These tests verify exact-span recovery from the beginning, middle, and end of documents; cross-document attribution when a generation contains text from multiple sources; negative cases with no valid match; full-document matching; and the correctness of summary statistics and exported span metadata. Together, these tests confirm that the tracing pipeline and its aggregate metrics behave deterministically under known conditions.

Second, we run end-to-end validations on both Common Pile and Dynaword using 25 sampled documents from each indexed corpus. For each document, we construct one full-document query and three 128-token partial queries anchored at the start, middle, and end, yielding 100 validation queries per corpus. A query is counted as a pass if SIMPLETRACE either retrieves the original source document or returns an exact span match that covers the query text. Tables 5 and 6 summarize the results.

The Common Pile validation shows near-perfect retrieval accuracy. Across all 100 queries, the source-document retrieval rate is 0.99 and the exact-text-match rate is 0.99, while the overall pass rate is 1.00. Full-document queries are recovered perfectly, with both document retrieval and exact text match rates equal to 1.00. For partial queries, middle and end windows are also recovered perfectly. The only deviation occurs for start-anchored partial queries, where the source-document retrieval and exact-text-match rates are 0.96 (24/25). However, even in that case the exact queried span is still recovered elsewhere in the corpus, giving a partial-span exact-query match rate of 1.00 and leaving the pass rate unchanged at 1.00.

This single missed source-document retrieval is consistent with the reported count of one partial query for which the original document ID was not returned despite an exact span match being found. In other words, the validation failure is not a failure to trace the text itself, but a failure to recover the specific originating document identifier in one duplicated or ambiguous case. We therefore view the Common Pile validation as evidence that SIMPLETRACE is reliable for both exact text attribution and downstream memorization measurement on large real-world corpora. Increasing the maximum number of documents that can be retrieved (now 10) will likely retrieve the exact document. From manual inspection of the result we noticed there are many documents (code) containing exactly the same text and so filling up easily the max 10 docs now retrieved.

We observe even stronger results on Dynaword, where all reported retrieval and exact-match metrics are perfect. Across all 100 Dynaword queries, the source-document retrieval rate is 1.00 and the exact-text-match rate is 1.00. The same holds for full-document queries and for all three partial-query settings (start, middle, and end), indicating perfect end-to-end recovery on the sampled set.

For completeness, the partial-span exact-query match rate on Common Pile is 1.00 for all three partial query types and 0.00 for full-document queries, as expected. No failed examples were logged there. Dynaword similarly logs no missing-document cases in the sampled validation set.

Table 5: End-to-end validation of SIMPLETRACE on Common Pile. Results are computed over 25 sampled documents, with one full-document query and three partial queries (start, middle, end) per document.

Query type	Doc ret.	Exact match	Pass
All queries	0.99	0.99	1.00
Full document	1.00	1.00	1.00
Partial start	0.96	0.96	1.00
Partial middle	1.00	1.00	1.00
Partial end	1.00	1.00	1.00

Table 6: End-to-end validation of SIMPLETRACE on Dynaword. Results are computed over 25 sampled documents, with one full-document query and three partial queries (start, middle, end) per document.

Query type	Doc ret.	Exact match
All queries	1.00	1.00
Full document	1.00	1.00
Partial start	1.00	1.00
Partial middle	1.00	1.00
Partial end	1.00	1.00

B Prompt Validation

The full results for prompt validation are presented in terms of different overlapping metrics in Figure 2. As wanted, the results show an increasing trend in overlapping across different prompt setting, so to better distinguish between propensity and capability scenarios.

C Memorization Across Training Stages

This appendix contains the full metric plots and detailed analysis for Experiments 3 and 4, which evaluate memorization across the three training stages of DFM Decoder on Dynaword and Common Pile respectively. The main finding – that memorization profiles are essentially stable across stages – is summarised in Section 5.3; the figures and extended discussion are provided here.

C.1 Memorization of Dynaword Across Training Stages

Figure 3 reports the core SIMPLETRACE metrics across the three training stages of the DFM Decoder model evaluated on Dynaword; Figure 4 reports the corresponding propensity scores.

Memorization is stable across training stages. All metrics are nearly identical across Stage 1, Stage 2, and the Final model within each prompt setting. ALS shows no variation across stages (15.68, 17.37, and 24.75 tokens for generic, specific, and prefix respectively). NVR and FMR likewise vary only minimally and show no directional trend. Span length distributions (Figure 5) are visually indistinguishable across stages for all three prompt settings: under non-adversarial prompts, matched spans are concentrated in the short (11–20 token) bucket with a sharp drop-off beyond 50 tokens, while prefix attacks produce a somewhat broader distribution reaching into the (51–100) bucket.

Propensity scores are similarly flat across stages. Generic NVR propensity ranges between 0.023 and 0.027 across stages, while specific FMR propensity stabilises at 0.125 from Stage 2 onward (Figure 4). Both values are substantially below the neutral score of 0.5, confirming that the low propensity observed for DFM Decoder on Dynaword is not an artefact of a particular checkpoint but a persistent characteristic throughout training.

Interpretation. The memorization profile of Dynaword content appears to be established early in training and does not intensify with continued pre-training. This is consistent with prior evidence that memorization is more likely for texts observed in later training steps (Kiyomaru et al., 2024): Common Pile data, seen in all stages, contributes a stable background signal, and the additional Dynaword exposure introduced during continual pre-training does not measurably increase the depth or rate of memorization beyond what is already present after Stage 1.

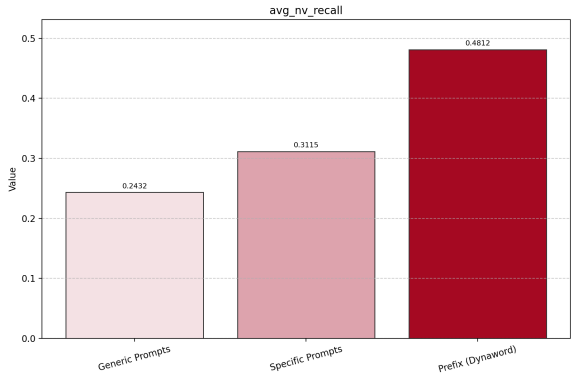
C.2 Exp. 4: Memorization of Common Pile Across Training Stages

Figure 6 reports the core SIMPLETRACE metrics across the three training stages of the DFM Decoder model evaluated on Common Pile; Figure 7 reports the corresponding propensity scores.

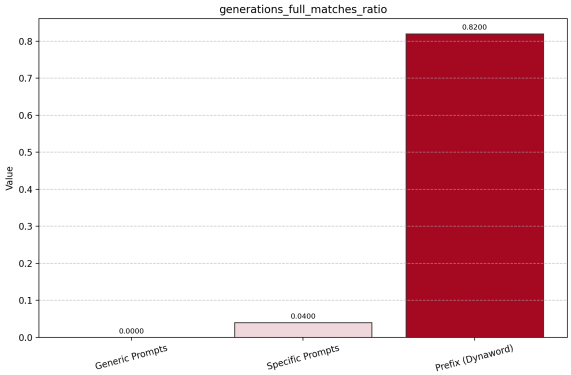
Memorization is stable across training stages. The memorization profile is essentially unchanged across Stage 1, Stage 2, and the Final model. ALS is identical across stages within each prompt setting: 23.57 (generic), 30.15 (specific), and 40.83 (prefix) tokens. NVR is also nearly flat: generic prompts remain at 0.0003, specific prompts vary only between 0.0092 and 0.0094, and prefix attacks vary between 0.0238 and 0.0243. FMR is 0.00 for all stages and all prompt settings, indicating no full verbatim reproductions of Common Pile content in any checkpoint of DFM Decoder.

Span length distributions are visually indistinguishable across stages. Figure 8 shows that the span length distributions for generic and specific prompts are concentrated mainly in the (7-10) and (11-20) token buckets throughout all three stages. Prefix attacks produce a broader tail toward longer spans – with some mass in the (21-50) and (51-100) buckets – but this shape is likewise unchanged across stages. The stability of the distribution confirms that continual pre-training on Dynaword data does not alter the depth or pattern of Common Pile memorization in DFM Decoder.

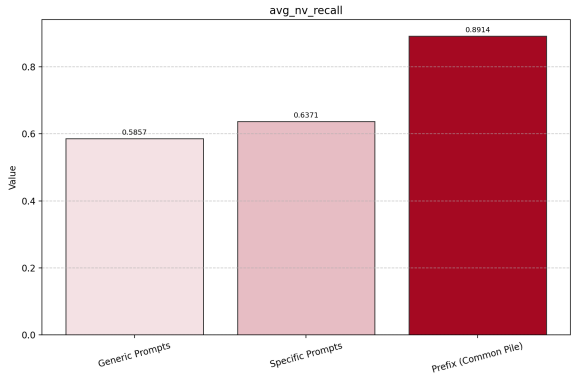
Propensity scores show the same stability. Generic NVR propensity remains around 0.013 across all three stages, while specific NVR propensity remains around 0.28 (Figure 7). Full-match propensity scores are 0.00 throughout, consistent with the absence of full verbatim reproductions in all settings. Both values are well below the neutral score of 0.5, indicating persistently low propensity to reproduce Common Pile content in non-



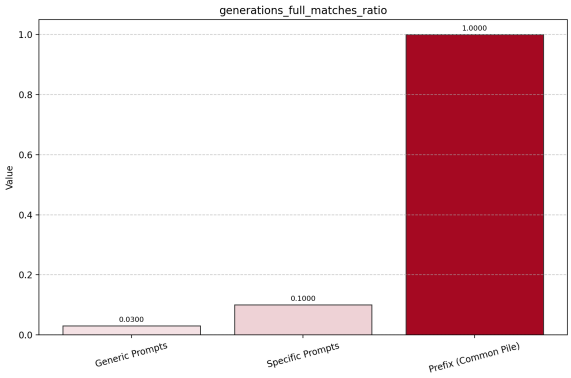
(a) Average near-verbatim recall between prompts and Dynaword.



(b) Fraction of prompts verbatim matched in Dynaword.

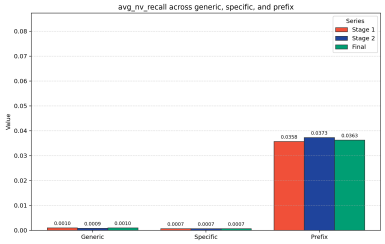


(c) Average near-verbatim recall between prompts and Common Pile.

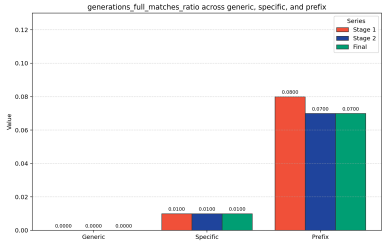


(d) Fraction of prompts verbatim matched in Common Pile.

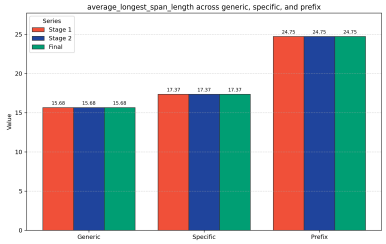
Figure 2: Evaluating overlapping between prompts and datasets across all prompt settings (Dynaword top, Common Pile bottom).



(a) Average near-verbatim recall per prompt setting and training stage.



(b) Fraction of generations with verbatim matches per training stage.



(c) Average longest span per training stage.

Figure 3: Memorization metrics for Dynaword across three training stages of the DFM Decoder model.

adversarial conditions. As noted in Section 5.3, this stability is consistent with Kiyomaru et al. (2024), who report a recency effect in memorization: the Common Pile signal is fixed by the end of Stage 1 and is neither amplified nor attenuated by the subsequent Dynaword-dominated continual pre-training stages.

D Span Length Distributions for Main Experiments

This appendix collects the span length distributions for Experiments 1, 2, 5, and 6, which were omitted from the main text for space. These figures provide a granular view of how verbatim overlaps between model generations and training documents are distributed across token-length buckets, complementing the aggregate metrics reported in Section 5.

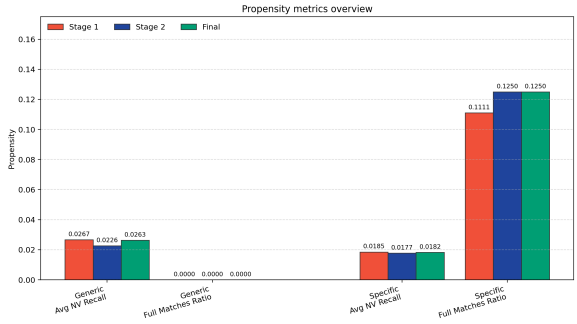


Figure 4: Propensity scores for Dynaword across training stages of the DFM Decoder model.

D.1 Dynaword Span Lengths in DFM Decoder

Figure 9 shows that under generic and specific prompts, matched spans are strongly concentrated in the short (11–20 token) bucket, with a sharp drop-off beyond 50 tokens and virtually no mass in the (101–150) or (151–∞) ranges. Under the prefix attack the distribution broadens noticeably: the (21–50) and (51–100) buckets gain a larger share, and a small but non-zero mass appears in the longest bucket, where the maximum matched span reaches 122 tokens. This confirms that prefix attacks increase not only the *rate* but also the *depth* of memorized reproduction.

D.2 Common Pile Span Lengths in Comma Model

Figure 10 shows that, across all three prompt settings, the (11–20) token bucket dominates. Nevertheless, the prefix attack distribution has noticeably more mass in longer buckets: approximately 23% of prefix-attack spans fall in the (21–50) range, versus 16% (generic) and 12% (specific). The prefix setting also has some presence in the (51–100) and (151–∞) buckets, which are largely absent for the non-adversarial settings. The overall shift toward longer spans under prefix attacks mirrors the pattern observed for Dynaword in Experiment 1, but the longer baseline spans for Common Pile reflect the greater verbatim overlap available in this larger, English corpus.

D.3 Common Pile vs. Dynaword Span Lengths (DFM Decoder)

Figure 11 compares span length distributions for the two corpora. Common Pile is shifted toward longer spans across all settings, with more mass in the 21–50 and longer buckets, especially under prefix prompting. Dynaword places more mass in shorter buckets, particularly below 10 tokens, reflecting the shorter average document length of this corpus relative to Common Pile. Both corpora show a broader distribution under prefix attacks, with the prefix-attack Dynaword distribution also gaining mass in the (21–50) bucket. Taken together, the two distributions suggest qualitatively different memorization profiles: Common Pile memorization tends to manifest as longer localized verbatim fragments, while Dynaword memorization produces shorter but occasionally complete generation-level reproductions (as evidenced by its higher `FMR` under prefix attacks; see Section 5.2).

D.4 Common Pile Span Lengths: Comma vs. DFM Decoder

Figure 12 shows that in the generic and specific settings both models concentrate most of their mass in the 7–10 and 11–20 token buckets, with additional mass in the 21–50 bucket. The key difference emerges under prefix attacks: Comma shifts more strongly toward longer spans, placing more mass than DFM Decoder in the 21–50, 51–100, and longest buckets, while DFM Decoder remains concentrated in the 11–20 bucket. This is consistent with Comma's higher capability-level memorization reported in Section 5.2 and further supports the interpretation that continual pre-training on Dynaword data partially attenuates the depth – though not the rate – of Common Pile memorization in DFM Decoder relative to its Comma base.

E Generating Prompt Settings

E.1 `Generic` Prompt Setting

For each of the following domains (comma separated) create 10 start of sentences prompts of various length.

Domains: `domains`

Format the output as a JSONL file like this: `example json`

Output the final JSONL in a code box

E.2 `Specific` Prompt Setting

Consider this dataset: `Dataset URL` and its domains: `domains`

For each domain create 10 start of sentences prompts of various length. You must not extract the prompts from the dataset but they should be somehow similar.

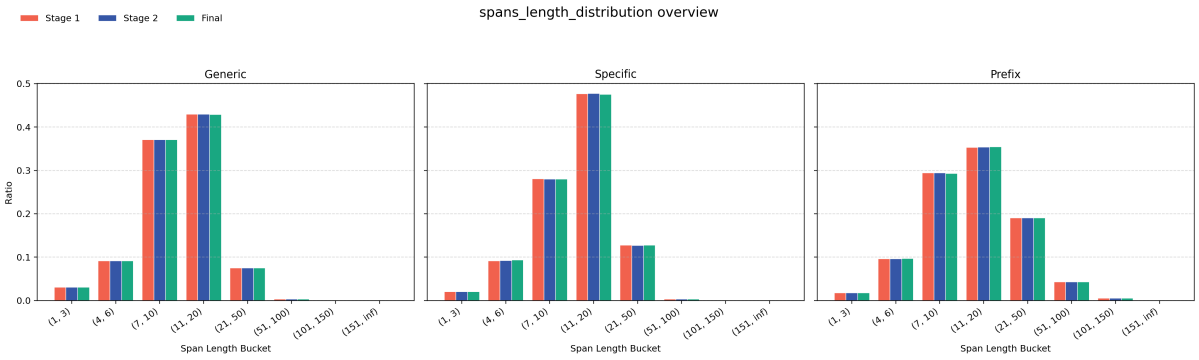
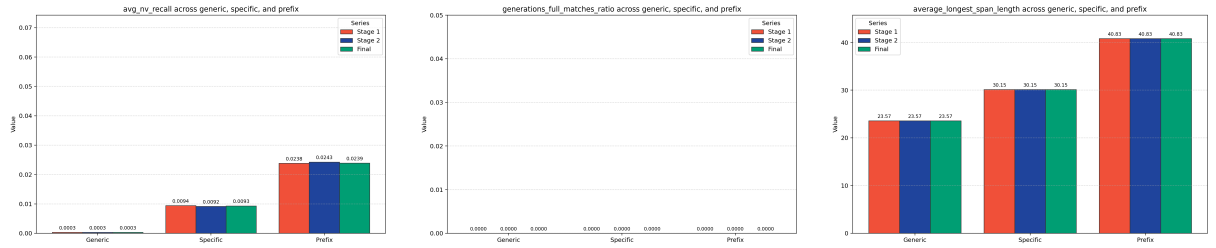


Figure 5: Span length distributions for Dynaword across training stages and prompt settings.



(a) Average near-verbatim recall per prompt setting and training stage.

(b) Fraction of generations with full verbatim matches per training stage.

(c) Average longest span per training stage.

Figure 6: Memorization metrics for Common Pile across three training stages of the DFM Decoder model.

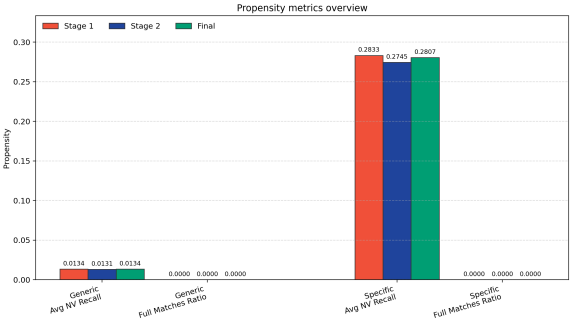


Figure 7: Propensity scores for Common Pile across training stages of the DFM Decoder model.

Format the output as a JSONL file like this: `example json`

Output the final JSONL in a code box

F Additional Memorization Metrics

k-Eidetic memorization (Carlini et al., 2021) defines a string s as memorized if it is extractable from f_θ and occurs in at most k training examples: $|\{x \in \mathcal{X} : s \subseteq x\}| \leq k$. **Near-duplication count** (Kiyomaru et al., 2024) counts training documents whose token-frequency vectors satisfy weighted Jaccard similarity $J_W(a, b) \geq 0.6$ with a generated span. **ROUGE-L** (Kassem et al., 2025) and **Token Accuracy** (Menta et al., 2025), the fraction of suffix tokens matching the greedy continuation, round out the set by capturing partial reproduction at different granularities.

G Use of AI Assistants

AI assistants were used only to support coding, grammar and style revisions, and literature search.

H SimpleTrace Metrics

SimpleTrace produces two kinds of outputs: per-document retrieval metrics attached to each traced span, and corpus-level summary metrics aggregated across all generations. Below, identifiers such as document ID lists and output paths are treated as metadata rather than metrics.

Per-document metrics.

`nv_recall` Near-verbatim recall between the generation and a retrieved document, defined as the fraction of generation words that reappear in the document as sufficiently long, aligned contiguous blocks.

`nv_matched_words` Number of generation words counted as part of near-verbatim matched blocks.

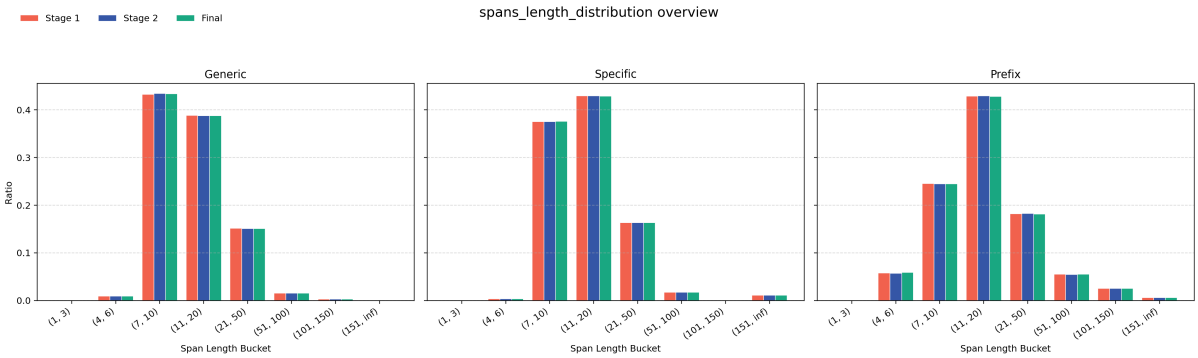


Figure 8: Span length distributions for Common Pile across training stages and prompt settings.

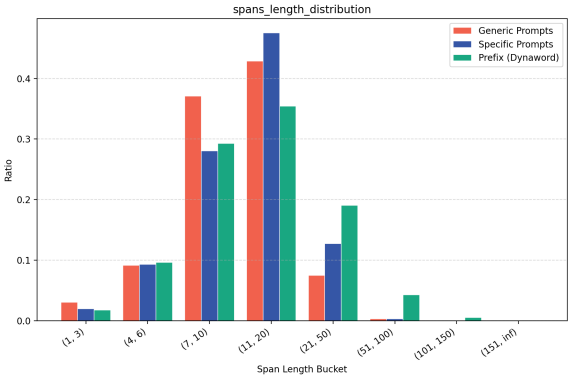


Figure 9: Span length distributions for Dynaword (DFM Decoder) across generic, specific, and prefix prompt settings. Spans are binned by token length; bars show the proportion of all matched spans falling in each bucket.

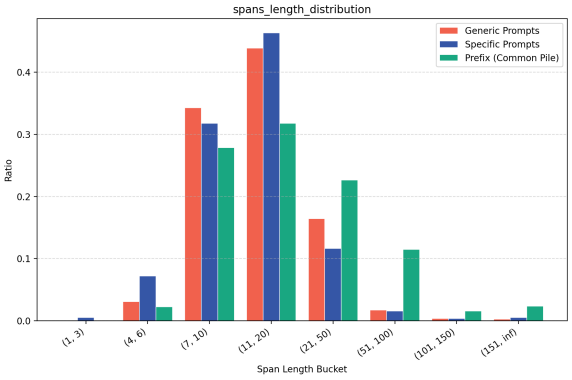


Figure 10: Span length distributions for Common Pile (Comma model) across generic, specific, and prefix prompt settings.

`nv_reference_words` Number of words in the generation, i.e. the denominator of `nv_recall`.

`nv_candidate_words` Number of words in the retrieved document.

`nv_missing_words` Number of generation words not covered by the near-verbatim match.

`nv_additional_words` Number of retrieved-document words not covered by the near-verbatim match.

Aggregate summary metrics.

`total_generations` Total number of generations evaluated.

`generations_with_spans` Number of generations for which SimpleTrace found at least one traced span.

`total_spans` Total number of final traced spans across all generations.

`average_longest_span_length` Average length of the longest traced span per generation.

`min_span_length` Shortest traced span length observed.

`max_span_length` Longest traced span length observed.

`n_token_span_ratio` Span-length threshold N used for the next metric.

`generations_with_n_token_span_ratio` Fraction of generations containing at least one span of length at least N.

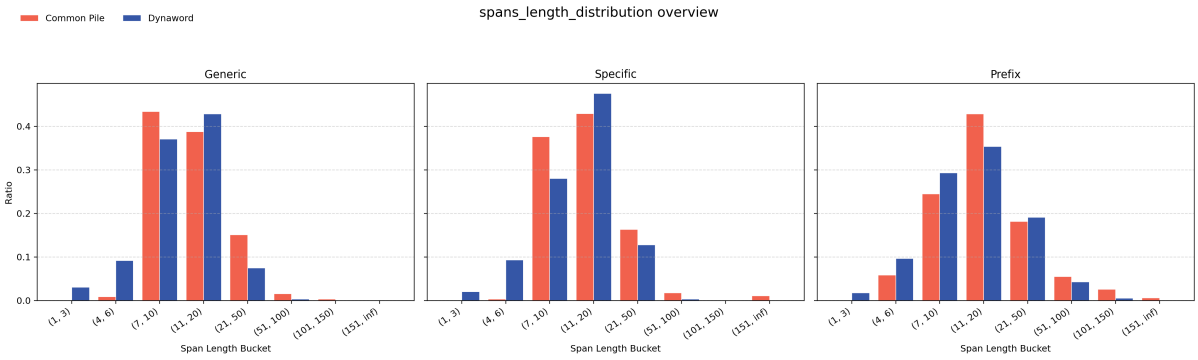


Figure 11: Span length distributions for Common Pile and Dynaword in DFM Decoder across generic, specific, and prefix prompt settings.

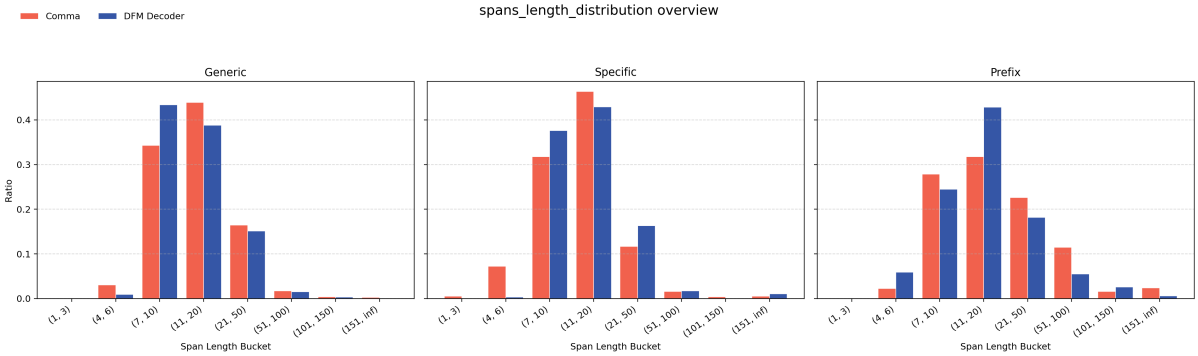


Figure 12: Span length distributions for Common Pile memorization in Comma and DFM Decoder across generic, specific, and prefix prompt settings.

`generations_full_matches_ratio` Fraction of generations with at least one retrieved document containing the full generation verbatim.

`generations_full_normalized_matches_ratio` Fraction of generations with at least one retrieved document containing the full generation after light normalization.

`total_docs` Total number of retrieved documents across all spans.

`unique_total_docs` Number of distinct retrieved documents.

`full_exact_matches` Number of retrieved documents containing the full generation verbatim.

`unique_full_matches` Number of distinct retrieved documents containing at least one full verbatim match.

`unique_full_matches_ratio` Ratio of distinct full-match documents to distinct retrieved documents.

`full_normalized_matches` Number of retrieved documents containing the full generation after normalization.

`unique_full_normalized_matches` Number of distinct retrieved documents with a normalized full match.

`unique_full_normalized_matches_ratio` Ratio of distinct normalized full-match documents to distinct retrieved documents.

`partial_matches` Number of retrieved documents with only partial span overlap rather than a full-generation match.

`unique_partial_matches` Number of distinct documents with at least one partial match.

`avg_nv_recall` Mean near-verbatim recall across all retrieved documents.

`max_nv_recall` Maximum near-verbatim recall observed among retrieved documents.

`docs_with_nv_recall` Number of retrieved documents with non-zero near-verbatim recall.

total_nv_matched_words Total number of near-verbatim matched words summed across retrieved documents.

generations_with_nv_recall Number of generations with at least one retrieved document having non-zero near-verbatim recall.

generations_with_nv_recall_ratio Fraction of generations with at least one retrieved document having non-zero near-verbatim recall.

nv_recall_threshold User-defined threshold used to flag especially strong near-verbatim matches.

generations_above_nv_recall_threshold Number of generations with at least one retrieved document whose `nv_recall` exceeds the threshold.

generations_above_nv_recall_threshold_ratio Fraction of generations with at least one retrieved document above the threshold.

docs_above_nv_recall_threshold Number of distinct retrieved documents whose `nv_recall` exceeds the threshold.

spans_length_counts_distribution Histogram of retrieved documents grouped by the token length of the matched span.

spans_length_distribution Normalized version of the previous histogram, reported as proportions.