DECOUPLING TOKEN ROLES IN AUTOREGRESSIVE PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive pretraining increasingly draws on heterogeneous data, making it important to understand how a model learns from an individual token. The next-token prediction objective naturally identifies a token's contribution with its own loss. However, each token is not only a prediction target but also context for what follows. Using controlled corruption, we decouple these two roles and find a reversal: making a noisy token easier to predict reduces its damage as a target but increases it as context. The same decoupling helps explain text generated by language models: generation selects each token by its fit to the prefix, while its role as context is never tested against an independently determined continuation, because that continuation is generated to fit it. At known corrupted positions, acting through the context can reduce damage that removing the token's own loss does not. Understanding and controlling what a model learns from a token therefore requires decoupling its roles.

1 INTRODUCTION

Autoregressive pretraining underlies current frontier language models (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023; Grattafiori et al., 2024; DeepSeek-AI, 2024; Yang et al., 2025) and draws on ever larger and more varied text. Scaling laws call for more training tokens as models grow (Kaplan et al., 2020; Hoffmann et al., 2022), while high-quality human-written text is limited (Villalobos et al., 2024; Muennighoff et al., 2023), so training corpora increasingly combine filtered web crawls (Penedo et al., 2024; Soldaini et al., 2024) with text generated by language models (Gunasekar et al., 2023; Maini et al., 2024; Ben Allal et al., 2025). Deciding what enters training has therefore become as important as how much data to use, and such decisions are now made at every level, from the mixture of sources (Xie et al., 2023; Liu et al., 2025) and the filtering of documents (Wenzek et al., 2020; Wettig et al., 2024; Li et al., 2024a) down to individual tokens (Lin et al., 2024; Hans et al., 2024; Rathi & Radford, 2026; Fang et al., 2025). At that finest level, deciding what enters training requires knowing what a model learns from an individual token in autoregressive pretraining.

The next-token objective suggests a simple account. The model is trained to predict each token x_i from its prefix $x_{<i}$, and the loss ℓ_i records how well it does so. Every loss term is attached to the token being predicted, so a token's contribution to training is naturally identified with its loss, and token-level methods that mask or reweight a position's loss act on this view (Lin et al., 2024; Fang et al., 2025; Helm et al., 2025). The same token, however, enters training a second time. Once predicted, x_i becomes context for every later prediction, and the model learns to use it. The consequences of this use appear in the losses of later tokens, never at position i. A token therefore takes part in training in two roles: in its *target role* the model learns to predict it, and in its *context role* the model learns to use it. In ordinary training the two roles always come together, because the same token is both predicted and used, and a token's loss describes only its target role.

To separate the two roles, we follow the use of corrupted labels to study generalization (Zhang et al., 2017) and use controlled corruption. We replace a fraction of the training tokens and route the same corruption, at the same positions, only into the prediction targets, only into the context, or into both. We then compare the trained models on clean held-out text. Since a token's loss is all that the objective records for it, we choose two kinds of replacement that differ sharply in their *own loss*, the loss at the noisy token's own position. *Random replacement* draws the noisy token from

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

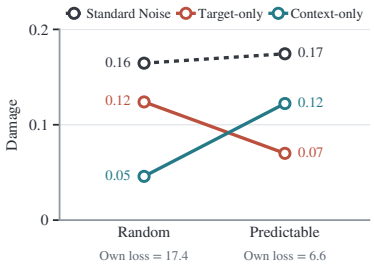


Figure 1: Despite very different own losses, random and predictable replacement cause similar damage when the corruption enters both roles. When it enters one role at a time, predictable replacement causes less damage than random replacement as a target and more as context. Damage is the increase in held-out cross entropy over clean training for GPT-2-large (Radford et al., 2019) trained on 9B tokens of OpenWebText (Gokaslan et al., 2019). Own losses are means over 128 matched positions under the clean model.

the vocabulary, so it is hard to predict from its prefix. *Predictable replacement* lets a small language model choose a noisy token that fits the same prefix, so it is much easier to predict.

We measure damage as the increase in held-out cross entropy over clean training. As shown in Figure 1, *Random* and *Predictable* replacement cause similar damage when the corruption enters both roles, even though their own losses differ sharply. Controlled corruption allows us to route the same noisy tokens into one role at a time and compare the two replacements within each role. When the corruption enters only the targets, the damage follows own loss: predictable replacement causes less damage than random replacement. When it enters only the context, the order reverses: predictable replacement causes more. This reversal holds for SmolLM2 (Ben Allal et al., 2025) trained from scratch and for continued pretraining of Qwen3 (Yang et al., 2025), across model sizes, training budgets, corruption rates and seeds.

To understand the reversal, we consider the two relations every token has with its text: to the prefix from which it is predicted, and to the continuation that is predicted from it. With controlled corruption we vary the two separately, choosing replacements whose fit to the prefix and fit to the continuation vary independently (Section 3.3). A better fit to the prefix lowers the damage as a target whether or not the continuation fits. As context, the same change lowers the damage when the continuation fits and raises it when it does not. The two roles thus respond to different properties of the same token. Understanding what a model learns from a token therefore requires decoupling its roles: the same token can serve training well as a target and poorly as context, or the reverse.

This understanding extends to text generated by a language model, which is also a form of predictable replacement: every generated token is chosen by its fit to the prefix, and *no token ever has to fit its future, because its future is generated to fit it*. We let a frozen model continue human text for 32 tokens and then return to the original human continuation, fixing what follows independently of the generated text. The more greedily the model decodes, the easier the generated text is to predict and the harder the first human token after it is to predict. Even random tokens disturb the fixed continuation less than generated text does under any decoding. Using generated text for pretraining therefore turns predictability from an observed property of the data into one that the generation process itself selects for, even though increasing it can work against the token's context role in learning.

Decoupling the roles also shows where token-level control of learning should act. Token-level methods mask or reweight a token's loss, which acts only on its target role. Even when the corrupted positions are known exactly, removing their losses does not reduce the damage of predictable replacement, which is more damaging as context. A reliability flag that marks the same positions in the input acts on the context role instead and recovers about 40% of the damage (Section 4.2). Which tokens to act on and which role to act through are therefore separate decisions.

Our main contributions can be summarized as follows:

1. Using controlled corruption routed into the targets, the context, or both, we show that making a noisy token easier to predict reduces its damage as a target and increases its damage as context.
2. Varying a noisy token's fit to its prefix and to its continuation separately, we show that damage as a target follows the fit to the prefix and that the continuation decides whether a better fit to the prefix lowers or raises the damage as context.
3. We show that autoregressive generation selects for the property that makes text easy to predict, while generating the continuation to fit those choices; when the continuation is fixed independently, increasing predictability makes the generated text fit it worse.
4. We show that token-level control should choose its role: knowing which tokens to act on does not determine whether to intervene through the target or context.

2 Related Work

Two roles in the objective. Influence functions separate a training token's influence as an output from its influence as an input (Grosse et al., 2023), the gradient at a position has been split into a term from its own loss and terms routed back from later positions (Liu et al., 2026), and state-prediction separation gives the two uses distinct representations (Monea et al., 2026). Corrupting only the targets of a language model raises its loss far less than the corruption rate (Peng & Jin, 2026), as does random noise in both roles (Ru et al., 2025). Corrupting only the context, with clean targets, is a long-standing regularizer (Xie et al., 2017; Bowman et al., 2016; Liu et al., 2022; Chen et al., 2026), and scheduled sampling likewise alters the context while keeping the targets (Bengio et al., 2015), its uniform-replacement control comparing random with model-sampled context by generation quality; the objective it induces is improper (Huszár, 2015). Replaced-token detection learns to find corrupted inputs (Clark et al., 2020), and translation studies separate source-side from target-side noise (Zhu et al., 2024). Other work assigns tokens to one role by design: prefix language models and prompt-loss weighting keep some tokens as context only (Wang et al., 2022; Shi et al., 2024; Huerta-Enochian & Ko, 2024), and continual pretraining on noised text drops noised words as targets (Kojima et al., 2025). Each of these acts on one role; comparing the two roles under the same corruption is what reveals the reversal.

Scores at the token's own position. Token-level selection and weighting score the predicted token by its excess loss over a reference model (Lin et al., 2024; Mindermann et al., 2022) or its predictive entropy (Su et al., 2024), truncate high-loss or high-error-norm tokens (Kang & Hashimoto, 2020; Li et al., 2024b), or clean tokens in instruction data (Pang et al., 2025); document-level pruning aggregates the same quantities into a perplexity (Wenzek et al., 2020; Marion et al., 2023; Ankner et al., 2025). Long-context scores compare the loss under a long and a short prefix (Fang et al., 2025; Helm et al., 2025), and context-length and attribution diagnostics measure how a prediction depends on the context it receives (Khandelwal et al., 2018; Mohebbi et al., 2023; Cohen-Wang et al., 2024). All of these score a token by how it is predicted, that is, through its target role.

Generated text. Truncated and greedy decoding concentrate on high-probability tokens (Fan et al., 2018; Holtzman et al., 2020), detectors exploit the resulting likelihood profile (Gehrmann et al., 2019; Mitchell et al., 2023), and a model learns text written by another model more readily because of its lower perplexity (Ren et al., 2024). Exposure-bias work feeds a model its own generated prefix and scores the continuation it then produces, which adapts to the prefix (Ranzato et al., 2016; He et al., 2021); later generated tokens are written to be consistent with earlier ones (Zhang et al., 2024). We fix the human continuation. Synthetic pretraining data are screened by likelihood or by judges (Gunasekar et al., 2023; Maini et al., 2024); recursive training on generated data collapses the distribution unless real data are retained (Shumailov et al., 2024; Gerstgrasser et al., 2024), more severely under narrow decoding (Dohmatob et al., 2024; Drayson et al., 2025), and noise consistent with its sequence harms more than local noise (Havrilla & Iyer, 2024). We separate what training learns from generated text as a target and as context.

Loss masks and markers. Loss masks remove selected target losses and keep the text as context, to limit memorization (Hans et al., 2024), to understand high-risk text without generating it (Wang et al., 2025), or to remove capabilities (Rathi & Radford, 2026); the masked identity remains encoded in the hidden states (Kosireddy & Lucas, 2026), and masking erroneous steps in synthetic reasoning data proves unnecessary when those steps are marked in the input (Ye et al., 2024). Markers condition the model on a tag attached to a sentence, segment or document, identifying back-translated, undesirable, junk or hazardous text (Caswell et al., 2019; Korbak et al., 2023; Allen-Zhu & Li, 2024; Lee et al., 2026). We compare the two at the same oracle positions, where the loss mask acts on the target role and a reliability flag on the context role.

3 The Two Roles Respond Differently to the Same Corruption

In this section we introduce noisy tokens into the training text and route the same corruption separately through the prediction targets and the context. In ordinary training the two roles cannot be observed apart, because the same token is always both predicted and used. This lets us measure how the same corrupted tokens affect learning when the model is trained to predict them and when it is trained to use them. We find a reversal between the two roles: corruption that is easier to predict does less

damage as a target but more as context. We then test how consistently it appears across training settings and which relation of the token, to its prefix or to its continuation, each role follows.

3.1 Decoupling the two token roles

For a text $x = (x_1, \ldots, x_T)$, autoregressive pretraining minimizes (Bengio et al., 2003; Radford et al., 2018)

$$\mathcal{L}(\theta) = \sum_{i=1}^{T} \ell_i(\theta), \qquad \ell_i(\theta) = -\log p_\theta(x_i \mid x_{<i}). \tag{1}$$

For a token x_k, ℓ_k measures how well the model predicts it as a *target* from its prefix. The same token also enters every later prediction through the *context* $x_{<i}$, $i > k$. The objective thus contains both roles, but the token's own loss describes only the one in which it is predicted.

We therefore compare the training consequences of the two roles: what happens when the model learns to predict a token and what happens when it learns to use the same token as context. To make the two roles explicit, we write the objective with a *context* stream c and a *target* stream y:

$$\mathcal{L}(\theta; c, y) = \sum_{i=1}^{T} -\log p_\theta(y_i \mid c_{<i}). \tag{2}$$

Ordinary training sets $c = y = x$, which returns Equation 1. The notation only separates the two ways in which a token enters training; it does not tell us what consequence each has for learning. To measure this, we change the token in one role while leaving it unchanged in the other, and compare the resulting models. As Zhang et al. (2017) corrupted labels and inputs to study generalization, we corrupt the text in a known way, which lets us study these consequences under controlled conditions: we choose which positions change and what replaces them. In a classifier the labels and the inputs are distinct; in autoregressive training every token is both a label and an input. The simplest controlled change is to replace the token. Unlike deletion or insertion, replacement keeps the length of the text and the position of every other token, so a corrupted stream stays aligned with a clean one. Let $\tilde{x}$ be a corrupted copy of x in which the tokens at a fixed set of positions are replaced. Since the two streams are written separately, the same corruption thus can enter either one while the other remains clean:

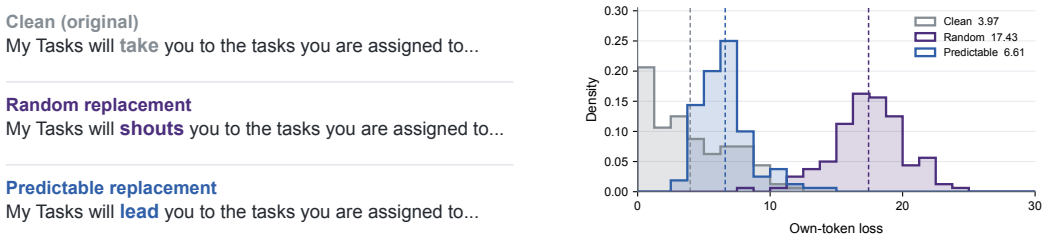
$$(c, y) = \begin{cases} (x, x), & \textit{Clean}, \\ (x, \tilde{x}), & \textit{Target-only Noise}, \\ (\tilde{x}, x), & \textit{Context-only Noise}, \\ (\tilde{x}, \tilde{x}), & \textit{Standard Noise}. \end{cases} \tag{3}$$

In *Target-only* training, a noisy token $\tilde{x}_k$ replaces the target at position k but never enters the context, where the original x_k remains. Only the prediction at position k changes: the model learns to predict the noisy token from clean text. In *Context-only* training, $\tilde{x}_k$ is never a prediction target, and x_k is still predicted at position k. The noisy token becomes context for the predictions that follow: the model learns to predict clean text after it. *Standard Noise* places the same noisy token in both roles, as ordinary training on corrupted text would. The three noisy conditions use the same corrupted positions and replacement tokens, together with the same training setup, so they differ only in the role through which the corruption enters.

3.2 Comparing the two token roles

With the two roles decoupled, differences between noisy tokens can be examined separately in each role. We begin with prediction difficulty, the quantity that a token's own loss records and that token-level selection acts on, so that the response of each role can be compared with the score ordinary training already provides. To compare tokens of very different difficulty at the same positions, we construct two corruptions that differ in how the replacement is chosen (Appendix A.4):

1. *Random replacement* draws uniformly from the vocabulary.
2. *Predictable replacement* selects a highly ranked whole-word candidate from a frozen language model given the same clean prefix.



(a) Replacement examples **(b)** Own loss

Figure 2: (a) One realized training position, with the same prefix and continuation for the original, random, and predictable tokens. (b) Own-loss distributions at 128 matched positions under a fixed clean GPT-2-large model; dashed lines mark the means.

In both cases the surrounding text is left unchanged, and neither choice looks at the continuation. Figure 2a shows the two replacements at one realized position. Across 128 matched positions, the mean own loss under a clean GPT-2-large model is 6.6 for predictable replacement, 17.4 for random replacement and 4.0 for the original tokens (Figure 2b). Predictable replacements are therefore much easier to predict as targets from the same clean prefix, though still harder than the tokens they replace.

We then train each corruption under the four conditions in Equation 3. We run this comparison with GPT-2-large (Radford et al., 2019) trained from scratch on OpenWebText (Gokaslan et al., 2019) using nanoGPT (Karpathy, 2022). Both corruptions target a 10% replacement rate, and within each corruption the same realized positions and replacement tokens are used across the three noisy routing conditions. The four conditions are trained under the matched setup above along the same 9B-token schedule and evaluated on the same clean held-out set, and training details in Appendix A.1.

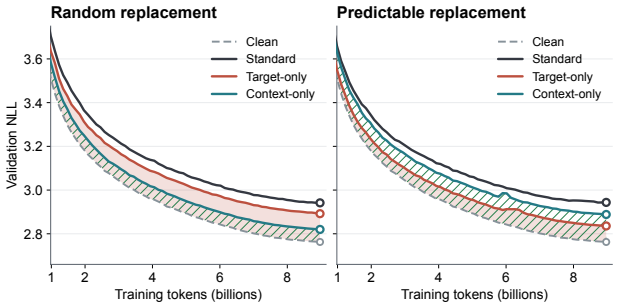


Figure 3: Throughout training, random and predictable replacement differ in which role is more damaging. Curves show held-out cross entropy during GPT-2 training, with damage given by the distance from the clean curve: target-only training is more damaging under random replacement, while context-only training is more damaging under predictable replacement.

Figure 3 shows the held-out cross entropy of each condition over training, one panel per kind of replacement. We measure the consequence of each condition by the increase in clean held-out cross entropy relative to matched clean training, which we call *damage*. Despite the large difference in the noisy tokens' own losses, random and predictable replacement cause similar damage under Standard Noise (Figure 3). Under target-only training, the difference follows own loss: predictable replacement causes less damage than random replacement. Under context-only training, however, the relation reverses: predictable replacement causes more damage. Damage under context-only training comes from using the noisy tokens to predict the unchanged continuation. The next-token objective assigns the losses for these predictions to the later tokens, while the noisy tokens' own losses describe how well the noisy tokens themselves are predicted. Under predictable replacement, this context-side damage even exceeds target-side damage throughout training. The advantage that a lower own loss gives a noisy token as a target therefore does not extend to its role as context.

The reversal holds across settings. We repeat the comparison in two other systems. SmolLM2 (Ben Allal et al., 2025) is trained from scratch on FineWeb-Edu (Lozhkov et al., 2024) with Nanotron (Hugging Face, 2023), which changes the architecture, the training text and the training framework. Qwen3 base models (Yang et al., 2025) are continued-pretrained on the same text with Hugging Face Transformers (Wolf et al., 2020), where the corruption enters a model that is already pretrained. For SmolLM2 we also vary model size from 135M to 1.7B parameters, the training budget from 1B to 10B tokens, and the corruption rate from 10% to 3% (Figure 4). Across these changes, predictable replacement remains less damaging than random replacement as a target and more damaging as context. At a 10% rate, the difference is 0.06 to 0.09 as a target and 0.04 to 0.07 as context. Training for 10B instead of 1B tokens leaves both differences in place, and lowering the rate

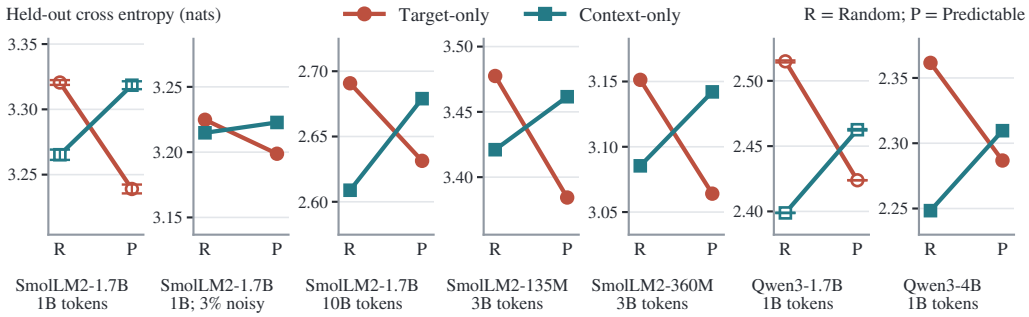
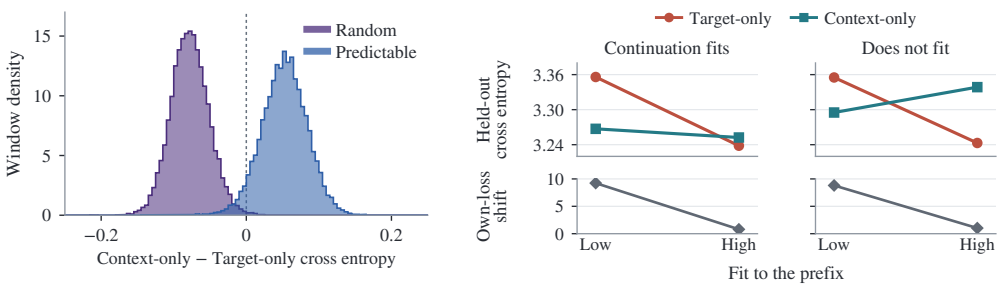


Figure 4: The reversal between the two roles is not specific to the initial GPT-2 comparison, and remains consistent across changes in model and training setup. Held-out cross entropy under target-only and context-only training for Random (R) and Predictable (P) replacement. The SmolLM2-1.7B and Qwen3-1.7B 1B-token settings each show the mean and observed range over three seeds.

to 3% shrinks them to 0.03 and 0.01 without changing their signs. We also repeat the SmolLM2-1.7B and Qwen3-1.7B 1B-token settings with three seeds; the reversal remains in every run.

3.3 Tracing the two token roles to the prefix and the continuation

The comparison between the target-only and context-only models in Section 3.2 holds on almost every held-out window. Across 19,531 unseen GPT-2 windows, the context-only model is worse than the target-only model in 95.1% of windows under predictable replacement and better in 99.6% under random replacement (Figure 5a). Section 3.2 located the context-side damage of predictable replacement in the predictions of the unchanged continuation. A replacement is predicted from its prefix and is used to predict its continuation. Random replacement controls neither relation, and predictable replacement is chosen for its fit to the prefix alone (Appendices A.4 and A.7), so the comparison leaves the fit to the continuation to chance. We therefore vary the two fits separately at the same positions.



(a) GPT-2-large, per held-out window

(b) SmolLM2-1.7B, four replacement sets

Figure 5: (a) On nearly every held-out window, the context-only model is worse than the target-only model under predictable replacement and better under random replacement. Each value is the context-only minus the target-only held-out cross entropy of GPT-2-large on one of 19,531 windows after 9B tokens; positive values mean that the context-only model is worse. (b) A better fit to the prefix lowers target-only cross entropy whether or not the continuation fits, and lowers context-only cross entropy only when it fits (top row); the own-loss shift δ of Equation 4 falls in both cases (bottom row). The four SmolLM2-1.7B replacement sets pair a low and a high fit to the prefix with a continuation that fits (left) or does not (right).

Varying the two relations. At each position, candidate replacements come from a common token-frequency group and are scored by a frozen language model on both relations: fit to the prefix by their probability under the clean prefix, and fit to the continuation by the loss on the clean continuation once the candidate is inserted. Four replacement sets pair a low and a high fit to the prefix with a continuation that fits and one that does not. We train each set under target-only and context-only training in the SmolLM2-1.7B setting of Figure 4 (Appendix A.5).

A better fit to the prefix lowers the target-only cross entropy by about the same amount whether or not the continuation fits (Figure 5b). The context-only model behaves differently: the same change

lowers its cross entropy slightly when the continuation fits and raises it when the continuation does not. Making a replacement easier to predict thus has a stable effect in its target role, while its effect as context changes direction with the continuation.

Among the four sets, the reversal of Section 3.2 thus appears only where the continuation does not fit. The comparison between the two models is the same in both columns: the target-only model is worse at low fit to the prefix and the context-only model at high fit, mainly because the damage as a target falls sharply as the replacement becomes easier to predict.

On the same sets we also measure the change in own loss when the clean target is replaced,

$$\delta_i = -\log p_\theta(\tilde{x}_i \mid x_{<i}) + \log p_\theta(x_i \mid x_{<i}), \tag{4}$$

under a fixed clean SmolLM2 checkpoint (Appendix A.6). The shift falls with fit to the prefix by nearly the same amount in both columns (Figure 5b, lower row). Own loss follows the fit to the prefix, as the target role does.

Decoupling token roles. The property that makes a token a better target can make it a worse context. Because every token plays both roles, their combined effect can hide what happens in each role, so understanding what a model learns from a token requires decoupling its roles.

4 Decoupling Token Roles in Practice

In this section we apply the decoupling to two practical settings. We first consider text generated by a language model, where every token is chosen for its fit to the prefix, the same relation that makes it easy to predict as a target. We test whether this also makes the generated text reliable as context for what follows (Section 4.1). We also use the two roles to examine how token-level interventions act when unreliable positions are known. We compare removing their losses with marking the same positions in the input, acting on the target and context roles respectively (Section 4.2).

4.1 From single corrupted tokens to generated text

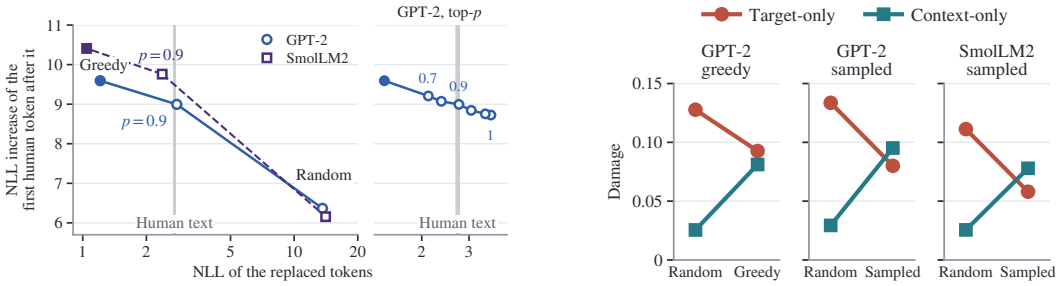
Predictable replacement uses a language model to choose a token that is easy to predict from its prefix. Autoregressive generation does the same repeatedly: at every step, next-token probabilities select what is written next, producing text that the model itself finds exceptionally easy to predict. Generated text is therefore a kind of *predictable replacement*. But this predictability comes entirely from the prefix, without regard to what comes after the generated text. Section 3.3 showed that making a token easier to predict does not have the same consequence in its two roles: it consistently helps as a target, while its effect as context depends on what follows. Generated text increasingly enters pretraining corpora (Gunasekar et al., 2023; Maini et al., 2024; Ben Allal et al., 2025), where these highly predictable tokens are both predicted and used as context.

We run the experiments with GPT-2 on OpenWebText and SmolLM2 on FineWeb-Edu, in a setup similar to Section 3.2. We select 10% of non-overlapping stretches of 32 consecutive tokens. For a stretch starting at position k, a frozen generator p_ϕ continues the human text,

$$\tilde{x}_t = \mathcal{D}\big(p_\phi(\cdot \mid \tilde{x}_{<t})\big), \qquad t = k, \ldots, k+31, \tag{5}$$

with $\tilde{x}_t = x_t$ at every other position, so that the original continuation resumes at $k+32$. The decoding rule $\mathcal{D}$ either takes the most probable token (*greedy* decoding) or samples from the smallest set of tokens whose probability reaches p (*top-p* sampling). Each generated token is thus conditioned on the human prefix and the tokens generated before it; the continuation $x_{\geq k+32}$ never enters Equation 5. A *random* condition places random tokens at the same positions.

Figure 6a places each replacement on the two relations of Section 3.3: its fit to the prefix, measured by the mean NLL of the replaced tokens under the generator, and its fit to the continuation, measured by the increase in NLL of the first human token after them. The second relation is normally absent during generation. Each generated token is chosen from its prefix, but the tokens that follow are then generated from a prefix containing that choice. Generation therefore chooses both the token and the trajectory along which it will subsequently be used as context. Our construction breaks this coupling after 32 tokens: generation stops and the original human continuation resumes. The generated text must now serve as context for a continuation that did not adapt to it. As decoding narrows from $p = 1$



(a) Fit to the prefix and to the continuation **(b)** Damage by role

Figure 6: (a) Fluent generated text disturbs the first human token after it more than random tokens do. Each point places one kind of replacement by the NLL of the replaced tokens under the generator (fit to the prefix) and the increase in NLL of the first human token after them (fit to the continuation), averaged over 20,000 fixed positions. Circles show GPT-2-medium on OpenWebText and squares SmolLM2-135M on FineWeb-Edu, each scored by its own generator; filled points are greedy decoding, and the band spans the mean NLL of the original human text under the two scorers. The right panel shows every top-p setting for GPT-2. (b) Relative to random tokens, generated text causes less damage under target-only training and more under context-only training. Each generated condition is paired with the Random and Clean runs of its own campaign.

to greedy, the two relations move apart. The NLL of the generated text falls from 3.5 to 1.2, while the increase at the first human token rises from 8.7 to 9.6. Random tokens give the contrast: despite an NLL of 13.6, they raise the NLL of the first human token by only 6.4, less than generated text under any decoding. SmolLM2 on FineWeb-Edu places random, sampled and greedy text in the same order (Appendix B). Narrower decoding therefore makes the generated trajectory easier to predict while making it harder to return to the fixed human continuation.

We next test whether this difference appears in training. Using top-p sampling with $p = 0.9$ and greedy decoding on GPT-2 and top-p sampling on SmolLM2, we train on generated and random text under the routing conditions of Equation 3. Relative to random tokens at the same positions, generated text causes less damage under target-only training and more under context-only training (Figure 6b). This holds for greedy and top-p generation on GPT-2 and for top-p generation on SmolLM2, where the two differences are about 0.05 each.

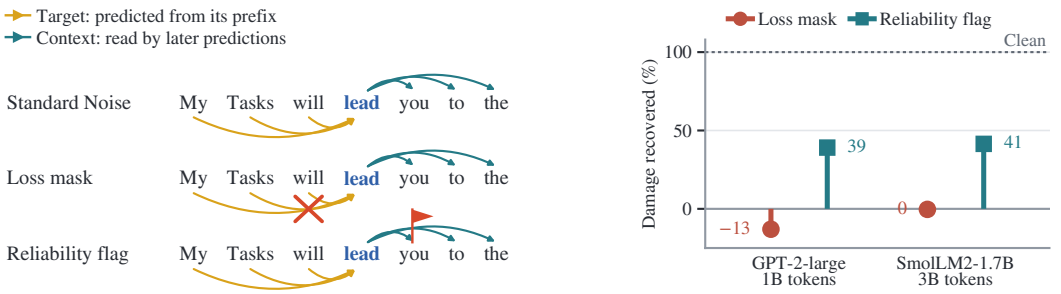
Together, these results reveal a difference between how generated text is produced and learned from. Autoregressive generation is unusual as a source of training data because the same mechanism used to learn from text is also used to produce the text to be learned from. In observed text, a token is constrained by both what precedes it and what follows. Generated text is exempt from this constraint: *it never has to fit its own continuation, because its continuation is generated to fit it*. Using autoregressive generation for training data therefore removes one of the constraints under which observed text is formed: compatibility with a continuation that does not adapt to the token.

4.2 Routing the same position labels to the target or the context

We next consider how to intervene on individual tokens during training. To isolate where an intervention should act, we provide oracle position labels, and take the corrupted positions of predictable replacement as known exactly. This leaves the question of what to do once those positions are known. The token at each position can be acted on through its own loss, where it is a target, or through the later predictions that read it as context. We compare the two with the same position labels. The comparison uses GPT-2 on OpenWebText and SmolLM2 on FineWeb-Edu (Appendix C), and measures each intervention by the fraction of the damage under standard noise that it recovers.

Loss masking. We begin with the intervention that existing token-level methods would take, removing the losses at the known positions (Lin et al., 2024; Hans et al., 2024; Wang et al., 2025). Let z be the observed stream and $m_i = 1$ mark a corrupted position. The masked objective is

$$\mathcal{L}_{\text{mask}}(z, m) = \sum_i (1 - m_i)\, \ell_i(z_i \mid z_{<i}). \tag{6}$$



(a) What each intervention changes **(b)** Damage recovered

Figure 7: (a) The loss mask and the reliability flag use the same position labels in different roles, shown on the example of Figure 2a. Yellow arrows mark the prediction of the replaced token *lead* from its prefix, and teal arrows the later predictions that read it. The mask removes the loss on *lead* and leaves it readable; the flag keeps the loss and marks *lead* for every later prediction that reads it. (b) At the oracle positions, the loss mask recovers none of the damage of predictable replacement and the flag about 40%. Damage recovered is the fraction of the standard noise damage removed, against each system's own clean and standard noise runs.

In the terms of Equation 3, the mask removes the noisy token from the target stream and leaves it in the context stream. In the example of Figure 2a, the model is no longer trained to predict *lead* after "My Tasks will", but it still reads *lead* when it predicts "you to the tasks" (Figure 7a). Unlike context-only routing, which supplies the clean target at a corrupted position, the mask supplies no target there, so it also removes about a tenth of all targets. The mask recovers none of the damage: the masked model ends slightly worse than the unmasked one on GPT-2 and level with it on SmolLM2, recovering -13% and -0.3% (Figure 7b), and on GPT-2 it is better on only 17% of the held-out windows. Knowing the right positions is therefore not enough when the intervention acts only on their target losses. The failure has two possible causes: the position labels may be of no use against this corruption, or the mask may deliver them to the wrong role. The latter is plausible because predictable replacement does more damage as context than as a target (Section 3.2), while masking its own loss leaves every later prediction to read it as before.

Reliability flag. To distinguish the two explanations, we route the same labels to the context instead. Tags in the input already mark the quality or origin of training text (Korbak et al., 2023; Caswell et al., 2019; Allen-Zhu & Li, 2024); we apply the same idea to single tokens. We keep every token and every loss and add the same label to the token's input representation,

$$h_i^{(0)} = e(z_i) + s\, r_{m_i}, \tag{7}$$

where r_0, r_1 are learned embeddings and s fixes their strength. Because z_i is predicted before it is read, its flag reaches only the predictions that use z_i as context. The flag thus leaves the noisy token in both streams and adds to the context stream a mark of which tokens are unreliable: the model is still trained to predict *lead*, and every later prediction reads *lead* together with its mark (Figure 7a). At evaluation every position is marked reliable. With $s = 0.5$, the flag recovers 39% of the damage on GPT-2 and 41% on SmolLM2 (Figure 7b), and on GPT-2 it lowers the loss on 97% of the held-out windows. The same position labels therefore help once later predictions are allowed to read them.

5 Conclusion

Autoregressive pretraining gives every training token two roles: the model learns to predict it as a target and to use it as context. By separating the two roles, we find that they can favor opposite tokens and follow different relations to the surrounding text. The loss at a position therefore captures only part of what the model learns from the token there. Generated text makes the limitation concrete: generation produces text that is exceptionally easy to predict without ensuring that it is equally reliable to learn from as context. Token-level control likewise requires choosing which role to act through. More broadly, learning from data at the token level requires accounting separately for learning a token and learning from it.

AI USE STATEMENT

Language models and AI agents were used for coding assistance, grammar polishing, and phrasing refinement during the drafting of this paper. While artificial intelligence assisted in the presentation, all remaining errors, oversights, and bugs remain the exclusive product of natural stupidity.

ETHICS STATEMENT

This work investigates foundational training dynamics and data-intervention mechanisms in autoregressive language models using publicly available pretraining corpora. Our study does not involve human subjects, private user data, or hazardous capabilities.

REPRODUCIBILITY STATEMENT

Experiments use GPT-2 (Radford et al., 2019), SmolLM2 (Ben Allal et al., 2025), and Qwen3 (Yang et al., 2025) on OpenWebText (Gokaslan et al., 2019) and FineWeb-Edu (Lozhkov et al., 2024). GPT-2 models are trained with nanoGPT (Karpathy, 2022), SmolLM2 models with Nanotron (Hugging Face, 2023), and Qwen3 models with Hugging Face Transformers (Wolf et al., 2020). All experiments run on NVIDIA GH200 GPUs, with at most four per job. We report model sizes, training budgets, corruption rates, and multi-seed results in the main text, and provide complete training, corruption, and evaluation settings with the submission.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *International Conference on Learning Representations*, 2025.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Isaac Caswell, Ciprian Chelba, and David Grangier. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation*, 2019.

Michael K Chen, Xikun Zhang, Fan Bai, Zhengding Hu, and Zhen Wang. Demystifying training-time augmentation for data-constrained language model pretraining. *arXiv preprint arXiv:2606.16246*, 2026.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing model generation to context. *arXiv preprint arXiv:2409.00729*, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. In *International Conference on Machine Learning*, 2024.

George Drayson, Emine Yilmaz, and Vasileios Lampos. Machine-generated text detection prevents language model collapse. *arXiv preprint arXiv:2502.15654*, 2025.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *International Conference on Learning Representations*, 2025.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2019.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A Roberts, Diyi Yang, David L Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus, 2019.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R Bowman. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Abhimanyu Hans, Yuxin Wen, Neel Jain, John Kirchenbauer, Hamid Kazemi, Prajwal Singhania, Siddharth Singh, Gowthami Somepalli, Jonas Geiping, Abhinav Bhatele, and Tom Goldstein. Be like a goldfish, don't memorize! mitigating memorization in generative language models. *arXiv preprint arXiv:2406.10209*, 2024.

Alex Havrilla and Maia Iyer. Understanding the effect of noise in LLM training data with algorithmic chains of thought. *arXiv preprint arXiv:2402.04004*, 2024.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James Glass. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Falko Helm, Nico Daheim, and Iryna Gurevych. Token weighting for long-range language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1440–1459, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Mathew Huerta-Enochian and Seung Yong Ko. Instruction fine-tuning: Does prompt loss matter? *arXiv preprint arXiv:2401.13586*, 2024.

Hugging Face. Nanotron: Minimalistic large language model 3D-parallelism training. https://github.com/huggingface/nanotron, 2023.

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

Daniel Kang and Tatsunori B Hashimoto. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

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

Andrej Karpathy. NanoGPT, 2022.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Continual pre-training on character-level noisy texts makes decoder-based language models robust few-shot learners. *Transactions of the Association for Computational Linguistics*, 13:831–847, 2025.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher Buckley, Jason Phang, Samuel R Bowman, and Ethan Perez. Pretraining language models with human preferences. In *International Conference on Machine Learning*, 2023.

Tagore Rao Kosireddy and Evan Lucas. Loss masking under the hood: Backdoor concealment and private data memorization in LLMs. In *Proceedings of the Sixth Workshop on Privacy in Natural Language Processing*, pp. 69–79, 2026.

Seunghyun Lee, Dongyoon Han, and Sangdoo Yun. Mark, don't erase: Token inoculation for dual-use knowledge in LLMs. *arXiv preprint arXiv:2607.18639*, 2026.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024a.

Tianjian Li, Haoran Xu, Philipp Koehn, Daniel Khashabi, and Kenton Murray. Error norm truncation: Robust training in the presence of data noise for text generation models. In *International Conference on Learning Representations*, 2024b.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*, 2024.

Hao Liu, Xinyang Geng, Lisa Lee, Igor Mordatch, Sergey Levine, Sharan Narang, and Pieter Abbeel. FCM: Forgetful causal masking makes causal language models better zero-shot learners. *arXiv preprint arXiv:2210.13432*, 2022.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. RegMix: Data mixture as regression for language model pre-training. In *International Conference on Learning Representations*, 2025.

Qing Liu, Ou Wu, and Yi Du. AlphaToken: Decoupling adaptation and stability for path-aware response token valuation in LLM post-training. *arXiv preprint arXiv:2606.01635*, 2026.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining LLMs at scale. *arXiv preprint arXiv:2309.04564*, 2023.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, 2022.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, 2023.

Hosein Mohebbi, Willem Zuidema, Grzegorz Chrupała, and Afra Alishahi. Quantifying context mixing in transformers. *arXiv preprint arXiv:2301.12971*, 2023.

Giovanni Monea, Nathan Godey, Kianté Brantley, and Yoav Artzi. The state-prediction separation hypothesis. *arXiv preprint arXiv:2607.01218*, 2026.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, 2023.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. Token cleaning: Fine-grained data selection for LLM supervised fine-tuning. In *International Conference on Machine Learning*, 2025.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Liu Peng and Yaochu Jin. Robustness of probabilistic models to low-quality data: A multi-perspective analysis. In *International Conference on Learning Representations*, 2026.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 2019.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Neil Rathi and Alec Radford. Shaping capabilities with token-level data filtering. *arXiv preprint arXiv:2601.21571*, 2026.

Xuan Ren, Biao Wu, and Lingqiao Liu. I learn better if you speak my language: Understanding the superior performance of fine-tuning large language models with LLM-generated responses. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Jinghan Ru, Yuxin Xie, Xianwei Zhuang, Yuguo Yin, Zhihui Guo, Zhiming Liu, Qianli Ren, and Yuexian Zou. Do we really have to filter out random noise in pre-training data for language models? *arXiv preprint arXiv:2502.06604*, 2025.

Zhengyan Shi, Adam X Yang, Bin Yu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. Instruction tuning with loss over instructions. In *Advances in Neural Information Processing Systems*, 2024.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 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*, 2024.

Zhenpeng Su, Xing Wu, Xue Bai, Zijia Lin, Hui Chen, Guiguang Ding, Wei Zhou, and Songlin Hu. MiLe loss: a new loss for mitigating the bias of learning difficulties in generative language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 250–262, 2024.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *International Conference on Machine Learning*, pp. 49523–49544, 2024.

Ryan Wang, Matthew Finlayson, Luca Soldaini, Swabha Swayamdipta, and Robin Jia. Teaching models to understand (but not generate) high-risk data. *arXiv preprint arXiv:2505.03052*, 2025.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective work best for zero-shot generalization? In *International Conference on Machine Learning*, 2022.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 4003–4012, 2020.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. QuRating: Selecting high-quality data for training language models. In *International Conference on Machine Learning*, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, 2023.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. Data noising as smoothing in neural network language models. In *International Conference on Learning Representations*, 2017.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems. *arXiv preprint arXiv:2408.16293*, 2024.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball. In *International Conference on Machine Learning*, 2024.

Dawei Zhu, Pinzhen Chen, Miaoran Zhang, Barry Haddow, Xiaoyu Shen, and Dietrich Klakow. Fine-tuning large language models to translate: Will a touch of noisy data in misaligned languages suffice? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 388–409, 2024.

A EXPERIMENTAL SETUP

Every trained model is evaluated with clean context and clean targets on held-out text that is disjoint, at the document level, from its training stream.

A.1 GPT-2-LARGE ON OPENWEBTEXT

This is the setup for the GPT-2 training comparison in Figure 3 and the held-out-window analysis in Figure 5a. The model has 773.5M parameters, 36 layers, 20 heads, hidden size 1,280, vocabulary size 50,257, and no bias or dropout. Training uses nanoGPT, bf16 compute, sequence length 1,024, and 491,520 predicted tokens per update. AdamW has $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, and gradient clipping 1.0. The peak learning rate is 2.5×10^{-4} with 1,000 warm-up updates and cosine decay to 2.5×10^{-5}. The 9B schedule has 18,311 updates; all role arms are compared at update 18,250, or 8,970,240,000 predicted tokens. The 1B runs of Section 4.2 use the same architecture and recipe and are compared at update 2,000, or 983,040,000 predicted tokens.

A.2 SMOLLM2 ON FINEWEB-EDU

This is the setup for the SmolLM2 comparisons in Figure 4 and the four-stream experiment in Figure 5b. SmolLM2-1.7B has 24 layers, hidden size 2,048, feed-forward size 8,192, 32 attention heads and a tied vocabulary of 49,152. It is initialized from scratch and trained with Nanotron in bf16, at sequence length 2,048 and 256 sequences per update (524,288 predicted tokens). AdamW uses $(0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 and gradient clipping 1.0. The 3B schedule has 5,722 updates, a peak learning rate of 2.5×10^{-4}, 1,000 warm-up updates, and cosine decay to 2.5×10^{-5}. The 1B family is trained separately with the same recipe and ends at update 1,907. The 10B schedule reuses the architecture, initialization, data order and corruption definitions, extends the cosine schedule to 19,064 updates, and consumes the full training stream without wrapping. The 135M and 360M models follow the 3B recipe. The flag comparison of Section 4.2 uses 5,719 updates (2,998,403,072 tokens) without wrapping and its own references (Appendix C).

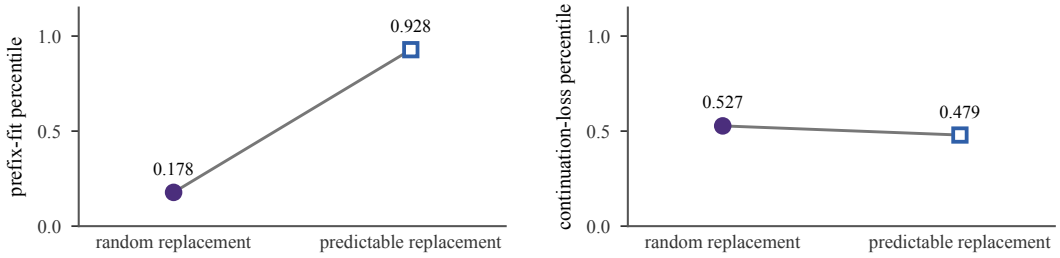
A.3 QWEN3 CONTINUED PRETRAINING

This is the setup for the Qwen3 continued-pretraining comparisons in Figure 4. Qwen3-4B-Base is continued-pretrained on FineWeb-Edu for 1,000 updates, with 512 sequences per global update, 2,048 predicted tokens per sequence, and 1.048576B predicted tokens in total. The fixed stream contains 512,000 non-overlapping chunks of 2,049 stored tokens and is not wrapped. Paired role arms share a frozen chunk permutation and replacement realization. Training uses fp32 master parameters and AdamW states, bf16 forward compute, fp32 reductions, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 on two-dimensional matrices and clipping 1.0. A 5% rewarm is followed by cosine decay to 10% of the peak learning rate. The peak, 4×10^{-5}, is the largest of 10^{-5}, 2×10^{-5} and 4×10^{-5} for which a 250-step probe on clean text kept the held-out cross entropy within 0.01 of the pretrained model's. Qwen3-1.7B-Base follows the same recipe. Predictable replacements are supplied by Qwen3-0.6B-Base from candidate ranks 1–41, which gives them the same separation from the original tokens under the pretrained model's own loss as ranks 10–50 give under a model trained from scratch.

A.4 CORRUPTION

The two replacements are built to differ in prediction difficulty and in nothing else that we control. Random replacement draws uniformly from the vocabulary, so the replacement bears no relation to the prefix. Predictable replacement takes a frozen small language model's distribution over the next token given the clean prefix and draws the replacement from the candidates ranked 10 to 50. The top ranks are excluded because they often contain the original token itself or a close variant of it, which would leave the text uncorrupted; ranks 10–50 give tokens that fit the prefix well but are still wrong, so their own loss lies between that of the original token and that of a random one (Figure 2b). Neither replacement sees the continuation, so any difference in fit to the continuation arises by chance.

Candidates for predictable replacement pass a complete-word filter. They must be space-initial and re-encode as a single token, so that the replacement occupies exactly the token it replaces and leaves



(a) Fit to the prefix **(b)** Fit to the continuation

Figure 8: Where the original Random and Predictable replacement families fall on the two relations. Values are mean within-position percentiles over 128 positions from each family.

the neighbouring tokens intact; alphabetic, at least four characters long and outside the stopword set, so that it is a content word rather than punctuation, a number or a function word; and inside a frequency-filtered English word list, so that it is a real word. The teacher is a pretrained GPT-2 (124M) for GPT-2 and SmolLM2-135M for SmolLM2, in each case given up to 512 clean tokens of left context. On SmolLM2 the word list contains 13,756 entries, and on average 27.6 of the 41 ranked candidates pass the filter, with a mean sampled rank of 30.5. Positions at which no candidate passes are left unchanged, so at a requested rate of 10% the realized rate is 9.766% for predictable replacement and 9.990% for random replacement.

A.5 Constructing the four relation streams

To separate the two relations, we construct four replacement streams that cross low and high fit to the prefix with low and high fit to the continuation. For each candidate, a frozen 135M teacher measures fit to the prefix by its log probability under the clean prefix. The same teacher then measures fit to the continuation by its mean loss on the next 16 unchanged tokens after the candidate is inserted; lower continuation loss means better fit.

At each position, we restrict candidates to a common token-frequency quintile. We take four accepted candidates from the top of the teacher's ranking and four accepted candidates drawn uniformly, giving two levels of fit to the prefix. Within each group, the candidates with the lowest and highest continuation loss supply the two continuation conditions. This produces the four cells of the relation map. The two scores are nearly uncorrelated within a position (median Spearman correlation -0.09 in a 128-position calibration), which allows the two relations to vary separately. Positions that cannot supply all four cells are skipped; fewer than 0.004% of positions are skipped in any shard.

All four streams use the same 102,437,240 replacement positions, a realized rate of 9.764%. Each stream is used once under target-only routing and once under context-only routing, giving eight training runs. Every run uses model seed 42 and follows the 3B-token SmolLM2-1.7B schedule for 2,000 updates, or 1.048576B training tokens.

A.6 Own loss under a clean model

We first check that changing fit to the prefix changes the replacement's own prediction loss. A clean SmolLM2-1.7B checkpoint trained for 3B tokens scores the original and replacement tokens under the same clean prefix. We evaluate all four relation-map streams at the same 10,000 positions, spread evenly through the corruption stream and chosen so that their 2,048-token prefixes do not overlap. At position i, we define the own-loss shift as

$$\delta_i = -\log p_\theta(\tilde{x}_i \mid x_{<i}) + \log p_\theta(x_i \mid x_{<i}).$$

This is the NLL of the replacement minus that of the original token, so positive values mean that the replacement is harder to predict. Figure 5b reports the mean shift for each stream.

A.7 WHERE THE ORIGINAL REPLACEMENTS FALL

The four streams above vary the two relations deliberately. We also locate the original Random and Predictable replacement families on the same axes. At 128 sampled positions from each family, the frozen 135M teacher scores fit to the clean prefix, and the clean SmolLM2-1.7B checkpoint trained for 3B tokens scores the unchanged next 16 tokens after each replacement. Because score scales differ across positions, we rank each realized replacement against alternative candidates at the same position and report its percentile. Predictable replacements lie near the top for fit to the prefix, with a mean percentile of 0.928, whereas random replacements lie near the bottom, at 0.178. Their continuation-loss percentiles are both near the middle, 0.479 and 0.527, and the bootstrap interval for their difference includes zero (Figure 8). The original families are therefore strongly separated by fit to the prefix, but neither was selected for fit to the continuation.

B GENERATED TEXT

This section gives the experimental details for Section 4.1. The fixed-model scores correspond to Figure 6a, while the training settings and Standard Noise results correspond to Figure 6b.

B.1 DATA AND DECODING

The GPT-2 experiments use a document-disjoint OpenWebText slice with 1.10B GPT-2 tokens for training and the next 20M tokens for validation. Starting at offset 17, the training stream is partitioned into stretches of 32 consecutive tokens; seed 1337 selects 10% of them without replacement, giving 3,437,497 stretches and 109,999,904 replaced positions. All corruption conditions use these positions and the same clean stream. A frozen pretrained GPT-2-medium receives up to 512 clean tokens to the left of each stretch and generates exactly 32 tokens, each conditioned on the tokens generated before it. End-of-text is suppressed during generation. Top-p sampling uses temperature 1, $p = 0.9$, and no top-k truncation. Greedy decoding takes the top-ranked token at every step. The random condition uses one seed-2024 uniform-vocabulary realization at the same positions. The original tokens outside each stretch, including the continuation on its right, are unchanged.

The GPT-2 training runs in Figure 6b use randomly initialized GPT-2-medium models with context length 1,024, a global batch of 262,144 tokens, and 4,000 updates (1.048576B tokens). AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, clipping 1.0, a 100-update linear warmup to 3×10^{-4}, and cosine decay to 3×10^{-5}. Runs within each comparison share the model seed, data order, positions, corruption realization, and evaluation procedure.

The SmolLM2 experiments use the SmolLM2 tokenizer and a document-disjoint FineWeb-Edu slice containing 1,099,997,390 training tokens and 19,999,369 validation tokens. A frozen SmolLM2-135M generates sampled text with $p = 0.9$ at the same kind of positions and, for scoring only, greedy text; the random condition uses the same positions. The training runs in Figure 6b use randomly initialized SmolLM2-135M models trained for 2,000 updates of 524,288 tokens (1.048576B tokens) with the recipe of Appendix A.2.

B.2 FIXED-MODEL SCORES: FIGURE 6A

We select 20,000 positions deterministically, excluding stretches that contain end-of-text, and score the replaced tokens and the first human token after them with a frozen model p_ϕ, the generator itself. For a stretch starting at position k, the fit to the prefix is the mean NLL of the replaced tokens,

$$-\frac{1}{32}\sum_{t=k}^{k+31}\log p_\phi(\tilde{x}_t \mid \tilde{x}_{<t}),$$

and the fit to the continuation is the increase in NLL of the first human token,

$$-\log p_\phi(x_{k+32} \mid \tilde{x}_{<k+32}) + \log p_\phi(x_{k+32} \mid x_{<k+32}).$$

Every sampled condition uses the same positions, prefixes, and sampling seed and generates 32 tokens; only p changes.

B.3 Standard Noise: Figure 6b

Under Standard Noise on GPT-2, the damage is 0.025 and 0.028 for random tokens in the greedy and sampled campaigns, 0.018 for greedy text and 0.009 for sampled text.

C Loss Mask and Reliability Flag

This section gives the settings for the single-token Predictable-replacement interventions in Section 4.2 and Figure 7b.

Damage recovered is $(V_{\mathrm{std}} - V_{\mathrm{op}})/(V_{\mathrm{std}} - V_{\mathrm{clean}})$, where V is held-out cross entropy and each system is compared with its own clean and Standard Noise runs; a negative value means that the intervention ends worse than Standard Noise.

GPT-2. The runs are compared at update $2{,}000$ of the 1B recipe (Appendix A.1). The reliability flag adds a two-state embedding, initialized to zero, to the plain backbone of the clean and Standard Noise runs. The same flag placed at randomly chosen positions at the same rate recovers none of the damage (-1%), so the gain of the oracle flag comes from the positions it marks rather than from the added embedding.

SmolLM2. Every run, including the clean and Standard Noise references, uses the backbone with the flag embedding, trained for $5{,}719$ updates of the 3B recipe (Appendix A.2); evaluation sets every flag to the reliable state. Because that state still adds the learned embedding $s\, r_0$ to every token, and because creating the embedding changes the initialization of the backbone, these runs form their own model family, and interventions are compared only within it.