MEASURING ORDER-LEARNABILITY VIA PREQUENTIAL CODELENGTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Does sequential data favor left-to-right prediction? Although entropy is invariant to factorization order, a finite learner may acquire some factorizations more efficiently than others. We measure order-learnability using prequential codelength, which accumulates the log-loss incurred when a model predicts each batch before training on it. After synthetic calibration, we compare 21 fixed orders across 13 corpora under a controlled Transformer training protocol. Our experiments reveal four findings. First, left-to-right is rarely uniquely favored: it is significantly better than every tested alternative only on mathematical chain-of-thought traces. Second, text is substantially more sensitive to local shuffling than to full reversal, suggesting that preserving local adjacency matters more than preserving forward direction for this learner. Third, the differences between orders change with training: both additional data and repeated passes narrow the relative gap on text but widen it on two protein sources. Fourth, frequency-sorted orders achieve substantially lower prediction loss on English Wikipedia while supplying additional information about the content. Our framework turns intuitions about a corpus's preferred prediction order into quantitative comparisons of learning efficiency under a fixed learner.

1 INTRODUCTION

Most language models factorize a sequence from left to right, predicting each token from its prefix. Any-order autoregressive models (Shih et al., 2022; Pannatier et al., 2024) and masked diffusion models (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) drop this commitment and learn to predict tokens in many orders. Two views of this choice coexist. On one, a fixed order is an assumption about the data that a sufficiently general learner should not need: a model trained over many orders can find whatever order structure the data carry, rather than having one imposed in advance. On the other, much data is produced sequentially—text is written word by word, and a step of reasoning builds on the steps before it—so the left-to-right factorization matches how the data arise, and a model that abandons it gives up a useful prior. Which view applies to a given corpus is a question about the data, and so far it has been argued largely from intuition.

Empirical evidence comes mainly from comparing model families. Masked diffusion has outperformed left-to-right models on planning and constraint-satisfaction tasks (Ye et al., 2025; Kim et al., 2025) and makes better use of repeated data, while left-to-right models remain more compute-efficient when data are not the constraint (Prabhudesai et al., 2025). Such comparisons, however, change the architecture, the training objective and the decoding procedure together with the generation order, so their outcomes do not isolate how a corpus responds to a change of order. Intuition does not isolate it either: it neither says how large a preference for left-to-right is nor distinguishes it from a broader sensitivity to which tokens are revealed together. What is missing is a measure: a quantity, computed for a corpus under a controlled learning protocol, of how strongly the corpus favors left-to-right and of what kind of order structure it rewards.

The entropy of the true distribution cannot distinguish factorization orders. By the chain rule, every ordering of a sequence factorizes the same joint distribution with the same total entropy, so an exact predictor codes the data equally well in every order (Luo et al., 2025). Which order a corpus favors therefore has no answer for the distribution alone. It acquires one for a learner of bounded capacity trained on finite data, which can find the conditionals induced by some orders easier to

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.

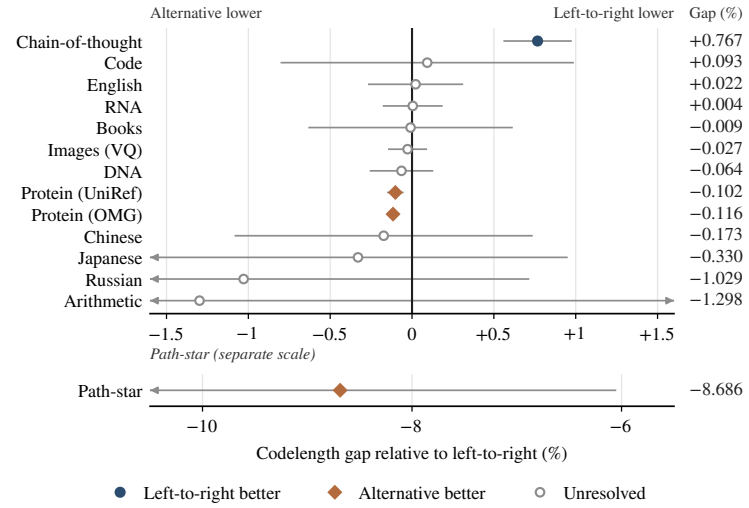

Figure 1: Codelength of the best tested alternative relative to left-to-right, averaged over 0.25M–100M tokens and eight paired runs, and expressed as a percentage of the left-to-right codelength. Positive values favor left-to-right; negative values favor an alternative. All point estimates except path-star have magnitude below 1.3%. Bars are 95% paired t-intervals. The best alternative is picked from all alternatives of the corpus after the data are seen, which makes its ordinary interval too optimistic; for negative margins the bars therefore use a Bonferroni-corrected interval, about twice as wide, that covers every alternative at once. Filled circles require positive intervals for every alternative; diamonds require the corrected interval to lie below zero; open circles indicate neither (Section 4.1). Arrowheads indicate truncated intervals; path-star uses a separate linear scale. Table 3 (Appendix D) names the best alternative and gives the margins in bits/token. VQ denotes vector quantization; corpus details are in Section 3.1.

learn than those induced by others (Appendix A). We call this property *order-learnability*: how efficiently a specified learner acquires a corpus under a given factorization order. Both the data and the learner are thus part of the quantity being measured. The answer also depends on how far learning has progressed: on English Wikipedia, reading backward is cheaper than reading forward early in training and more expensive later (Section 2.3), so a comparison at a single budget can reverse the verdict.

We introduce a calibrated measure of order-learnability based on *prequential codelength*, drawing on the connection between prediction and lossless compression in minimum description length (MDL) (Blier & Ollivier, 2018; Bornschein et al., 2023). For each order, a reference model learns the corpus sequentially: it predicts each batch using only what it has learned from previous batches, records its log-loss, and then updates its parameters. Summing these losses gives the number of bits spent encoding the corpus while learning it. This score has an operational interpretation: given an agreed learning protocol, a receiver can reconstruct the successive predictors from data already decoded, without being sent the learned parameters. An order that enables more accurate predictions earlier yields a shorter code. Differences between orders consequently measure, in bits, how much less efficiently one order is learned than another. Comparing left-to-right with a specified set of alternative orders then measures how strongly the corpus favors left-to-right under the specified learner and training budget (Section 2).

Every order is learned by the same architecture under a matched training protocol, with its own tuned learning rate. This compares orders under a common architecture and a shared learning-rate selection procedure. On synthetic data the measure behaves as it should: it shows no preference among orders when the data carry no order structure, and it identifies the most learnable order when the data are built to favor one (Section 3.4). We then apply it to thirteen corpora spanning text, code, biological sequences, images and synthetic tasks, with a second source for protein. We use a common set of twenty-one orders, with additional orders for images and path-star. Because the advantage of one order over another can change as learning proceeds, we compare orders over the whole course of training rather than only at its end (Section 2.3).

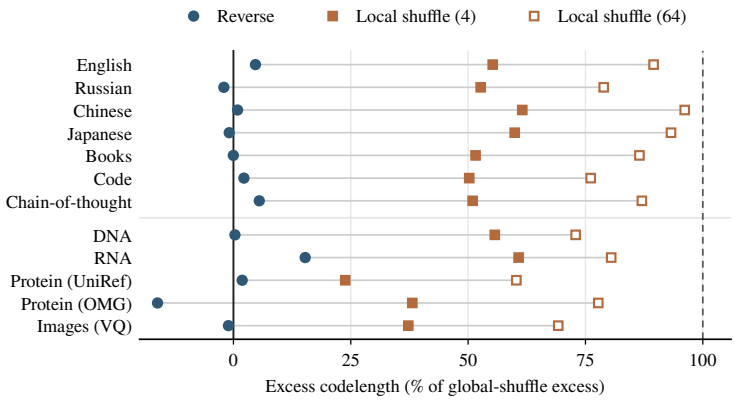

Figure 2: Excess prequential codelength of reversal and local shuffling relative to left-to-right, normalized by the excess codelength of a global shuffle on the same corpus. Zero matches left-to-right, 100 matches global shuffle, and negative values indicate a lower codelength than left-to-right. Reversal preserves adjacency; local shuffling permutes tokens within blocks of 4 or 64 while retaining the order of the blocks. On the seven text corpora above the separator, reversal changes the codelength by at most 5.5% of the global-shuffle excess, whereas four-token shuffling adds 50–62%. Points are means over eight paired runs, averaged over 0.25M–100M tokens. Arithmetic and path-star, which do not follow this pattern, are given in Table 4.

The experiments reveal four main findings. First, the point estimates of the gap between left-to-right and its best tested alternative have magnitude below 1.3% of the left-to-right codelength on every corpus except path-star (Figure 1). Left-to-right is significantly better than every alternative only on chain-of-thought traces. On both protein sources an alternative is significantly better, though by only about 0.1%, and path-star clearly disfavors left-to-right: full reversal codes it in 8.7% fewer bits. On the remaining corpora, neither left-to-right nor its best alternative is significantly better (Section 4.1).

Second, the text corpora are much more sensitive to local shuffling than to reversal (Figure 2). Reading backward keeps originally adjacent tokens together while reversing their direction. Shuffling within blocks preserves the order of the blocks but can separate neighboring tokens inside each block. On the seven text corpora, reversal changes the codelength by at most 5.5% of the excess codelength of a global shuffle, whereas shuffling within blocks of four adds 50–62%. This contrast indicates that preserving local adjacency accounts for much more of the advantage over a global shuffle than preserving the left-to-right direction (Section 4.2).

Third, multi-epoch training reveals that differences in order-learnability also depend on the stage of training. With further passes over the same data, the gap between orders shrinks on all seven text corpora, as the orders that start behind catch up, but grows on both protein sources, where differences emerge only as the model learns more (Section 4.3). Measured early in training, order effects are therefore overstated on text and understated on protein.

Fourth, choosing an order from the tokens themselves also conveys information about their content. Frequency sorting illustrates how this additional signal can make English Wikipedia easier to predict (Section 4.4).

Section 2 defines the measure, Section 3 describes the corpora, orders and training protocol and calibrates the measure on synthetic data, and Section 4 reports the findings in turn; Section 5 reviews related work and Section 6 discusses limitations.

2 Measuring order-learnability

We want to know how efficiently a given learner acquires a corpus when the positions of each window are predicted in different orders. The true distribution cannot answer this, since by the chain rule it assigns each window the same probability in every order; what can differ is how many bits the learner spends to approach it. A coding game makes this precise.

2.1 A TRANSMISSION GAME

Suppose Alice sends a corpus to Bob with an arithmetic coder. If both knew the true distribution p, each window x would cost $-\log_2 p(x)$ bits in whatever order its positions were sent, since the chain rule factorizes p in every order. Without p, they start from the same untrained network and a shared training recipe: Alice codes each batch with the current network, then both train on it, so Bob can decode every batch without the network ever being sent. The total codelength is the ideal codelength of the data plus a surcharge for what the network has not yet learned, which shrinks as it learns.

Now play the game twice on the same batches from the same initialization, changing only the order in which positions are predicted (and the learning rate tuned for it). The ideal codelength is the same in both games, so the difference in total codelength is exactly the difference in surcharge: the extra bits one order spends to learn the corpus. This is what we measure. An order that pays less is more learnable for this learner and this amount of data, and the entropy of the data, unknown for any natural corpus, never has to be estimated.

2.2 PREQUENTIAL CODELENGTH

The total codelength of this game is the prequential codelength (Blier & Ollivier, 2018), here with the model class fixed and the order varied. Let $x = (x_0, \dots, x_{n-1})$ be a window of n tokens, and let $(\sigma(0), \dots, \sigma(n-1))$ list its positions $0, \dots, n-1$ in prediction order. At step $t \in \{0, \dots, n-1\}$, the revealed (position, token) pairs are $r_t = \{(\sigma(s), x_{\sigma(s)}) : 0 \le s < t\}$. A network with parameters θ reading in order σ defines $q_{\theta,\sigma}(x) = \prod_{t=0}^{n-1} q_\theta(x_{\sigma(t)} \mid \sigma(t), r_t)$. With batches $D_1, D_2, \dots$ of windows, $q(D_b) = \prod_{x \in D_b} q(x)$, parameters $\theta_{<b}$ after training on the first $b-1$ batches, and sums over the batches that make up the first N tokens,

$$L_\sigma(N) \;=\; \sum_b -\log_2 q_{\theta_{<b},\sigma}(D_b) \;=\; \underbrace{\sum_b -\log_2 p(D_b)}_{C(N)} \;+\; \underbrace{\sum_b \log_2 \frac{p(D_b)}{q_{\theta_{<b},\sigma}(D_b)}}_{R_\sigma(N)} \tag{1}$$

bits. Its increment over one batch, per token, is the per-token prequential codelength of that batch: for the batch D_b that contains token N, $\ell_\sigma(N) = -\log_2 q_{\theta_{<b},\sigma}(D_b)/(n\,|D_b|)$, where $|D_b|$ counts windows, giving the loss of the current model on a batch it has not yet been trained on. Throughout, L denotes a total codelength in bits and ℓ a codelength per token. The ideal codelength $C(N)$ is the same for every order by the chain rule (Appendix A); its expectation is the entropy of the N tokens coded. The surcharge $R_\sigma(N)$ is the redundancy of the code, with nonnegative expectation $\sum_b \mathbb{E}\,\mathrm{KL}(p_b \,\|\, q_{\theta_{<b},\sigma})$, where p_b is the distribution of a batch and KL denotes Kullback–Leibler divergence. Since any two orders σ and τ code the same batches (Section 3.3),

$$L_\sigma(N) - L_\tau(N) \;=\; R_\sigma(N) - R_\tau(N) \tag{2}$$

holds exactly, not only in expectation, and σ is more learnable than τ at budget N when $L_\sigma(N) < L_\tau(N)$.

2.3 COMPARING ORDERS OVER A RANGE OF BUDGETS

The codelength per token at a single budget says how well an order predicts at that point, not how many bits it took to get there, and the difference between two orders can change sign along the curve: on English Wikipedia, reading in reverse has a codelength 0.1377 ($[0.0409, 0.2345]$) bits/token lower than left-to-right at 6.25M tokens and 0.0831 ($[0.0439, 0.1222]$) higher at 100M. We therefore compare orders by the codelength of a whole range of budgets $[a, b]$, per token,

$$\bar{\ell}_\sigma \;=\; \frac{L_\sigma(b) - L_\sigma(a)}{b-a} \;=\; \frac{1}{b-a}\int_a^b \ell_\sigma(N)\, dN,$$

and call $\bar{\ell}_\sigma - \bar{\ell}_{\mathrm{id}}$ the *excess codelength* of σ (bits/token), where id denotes the left-to-right order. It weights every token in the range equally and needs neither a fit nor smoothing, and by equation 2 it is exactly the difference in surcharge per token between σ and identity over the range. We take $[a, b] = [0.25\mathrm{M}, 100\mathrm{M}]$, starting where the learning-rate warm-up ends (Section 3.3).

Where a single budget is read, as above, or a curve is drawn, we smooth ℓ_σ with an exponential moving average over batches, $\tilde{\ell}_\sigma(N)$, with a half-life of 32 batches (about 1M tokens), because the codelength of a single batch of 32 documents is far noisier than most differences between orders. The full curves are in Figure 11.

3 Experimental setup

3.1 Corpora

We use thirteen corpora, eleven collected and two generated. The collected corpora are Wikipedia in English, Japanese, Russian and Chinese (Wikimedia Foundation, 2023), written independently in each language rather than translated; pre-1929 books (Kandpal et al., 2025); Python code (Fujii et al., 2026); mathematical chain-of-thought traces (Moshkov et al., 2025); human DNA (Schneider et al., 2017); RNA (RNAcentral Consortium, 2021); protein; and ImageNet images tokenized by a VQ codebook (Deng et al., 2009; Patil et al., 2024). Protein is drawn from two sources, UniRef50 (Suzek et al., 2015) and the metagenomic OMG (Cornman et al., 2025), and counted as one corpus, so the tables and figures have fourteen rows. The generated corpora are path-star graphs (Bachmann & Nagarajan, 2024) and long multiplication of eight-digit numbers. Each document contributes a single window of 1024 tokens (Section 3.3), except that a path-star instance fills 268. Every corpus has a training pool of 10^8 tokens, 10M held-out tokens for evaluation (5M on books) and 20M for learning-rate tuning, disjoint by document; Appendix B gives the source, tokenization and definition of a document for each (Table 1). Because the structure the model extracts beyond unigram frequencies ranges from 7.40 bits/token on code to 0.11 on RNA (Table 6), every cross-corpus comparison in Section 4 is made in dimensionless terms.

3.2 Orders

We compare twenty-one orders from ten families (Appendix G, Figure 6): left-to-right, or identity (`id`); reverse (`rev`); local shuffle `ls`$\langle w \rangle$ within blocks of $w \in \{4, 8, 16, 32, 64\}$; skeleton infill `sk`$\langle b \rangle$, every b-th position first; dyadic refinement (`skd64`), its hierarchical variant starting at stride 64; evens-then-odds (`eo`), its $b = 2$ case; fill-in-middle `fim`$\langle p \rangle$, which defers the middle $p\%$ of the window until after the prefix and suffix; goal-first `gf`$\langle g \rangle$, the last g positions first; round-robin `rr`$\langle k \rangle$ across k segments; and a global shuffle (`global0`). Numeric suffixes specify the parameter: `ls4` means `ls`$\langle 4 \rangle$, and `fim50` means `fim`$\langle 50 \rangle$. Each is a single permutation, fixed before training and shared by all windows, so `global0` is not the per-window random order of any-order training. The image and graph corpora add orders of their own (Appendix I). Two content-dependent orders, which sort each window by token frequency, are compared separately in Section 4.4.

3.3 Learner and training

Every order is learned by the same σ-GPT-style decoder (Pannatier et al., 2024): at each step, its input concatenates the embedding of the current token with encodings of that token's position in the original window and of the position to be predicted next, so it receives exactly what an order reveals, r_t and $\sigma(t)$. It has 17.1M parameters at $V = 8192$ and about 11M at smaller V (Appendix B).

We train with AdamW (Loshchilov & Hutter, 2019) ($(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.01, gradient clipping at 1.0), a linear warm-up over 0.25M tokens and a constant rate thereafter, so that the codelength at N does not depend on the planned training length. Each corpus is run in eight replicates with different seeds within corpora but the same across corpora. The learning rate of each (corpus, order) pair is selected from a grid of powers of two on a split disjoint from the training and evaluation pools, with the optimum required to lie inside the grid, and is then fixed across replicates (Appendix J). Within a replicate, all orders share the initial weights, the document order and the batch boundaries, so $C(N)$ is common to all of them; intervals are paired t-intervals over the replicates, adjusted for multiplicity where an order is selected after the fact (Section 4.1). Each document contributes one window, at a random offset, and all orders of a corpus share one tokenizer (Appendix B).[1]

[1]Each run uses a single H100. The single-pass curves of all corpora and the ten calibration anchors take about 37 GPU-hours, the convergence experiment about 100, and the learning-rate sweeps about 5.

3.4 CALIBRATING THE INSTRUMENT

Before using the measure to compare corpora, we check that it behaves correctly in two cases where the right answer is known: it should show no preference among orders when the data have no order structure, and it should single out the right order when that order is designed to be optimal. We test both on synthetic anchors of 100M tokens over a vocabulary of 64 symbols, each run in eight replicates. The instrument passes every test (Table 2, Appendix C).

No preference without structure. In the iid anchor, every token is drawn uniformly and independently, so every order has the same optimal codelength, 6 bits/token, and none can be learned more efficiently than another. Across ten orders, the measured codelengths agree to within 1.4×10^{-5} bits/token, about two parts in a million, and no order differs significantly from identity. This range (maximum minus minimum) is more than two orders of magnitude smaller than any margin we report as significant in Section 4.

The planted order is found. Each of the other nine anchors plants a target order σ^*. We generate a Markov chain in which each token is a fixed function of the previous one with probability 0.95 and is otherwise drawn uniformly, and write it into the window along σ^*, so that reading in σ^* recovers the chain. All orders have the same entropy (Appendix C), but only σ^* reveals every token easily after the one it depends on; most other orders scatter this dependence across the window. The nine targets cover every family other than identity: `global0`, `rev`, `ls64`, `rr8`, `eo`, `skd64`, `sk4`, `fim50` and `gf32`. On every anchor the instrument measures σ^* as the most learnable order, with a margin over the runner-up whose 95% interval excludes zero. The margin ranges from 1.5258 bits/token on the global-shuffle anchor, where every other order separates nearly all consecutive tokens of the chain, to 0.0013 on the reversal anchor, where identity keeps every adjacency and differs from σ^* only in direction.

4 DO REAL CORPORA FAVOR LEFT-TO-RIGHT?

We ask four empirical questions: whether left-to-right is the most learnable tested order, whether text is more sensitive to reversal or local shuffling, how the differences change with training, and how content-dependent orders compare with fixed ones.

4.1 LEFT-TO-RIGHT IS RARELY PRIVILEGED

On most corpora, the estimated codelength difference between left-to-right and its best tested alternative is small. We measure its advantage by the margin

$$m \;=\; \min_{\sigma \neq \mathrm{id}} \big(\bar{\ell}_\sigma - \bar{\ell}_{\mathrm{id}}\big),$$

the excess codelength (Section 2.3) of the best alternative, so that a positive margin means that no tested order has a lower codelength than left-to-right. Left-to-right is significantly best if the paired 95% interval of every alternative lies above zero, which needs no adjustment because all of these claims must hold at once (Berger, 1982). An alternative is significantly better only if the Bonferroni-adjusted interval of the best alternative lies below zero, because that order is picked after the data are seen (Appendix D).

Except on path-star, the point estimate of the margin has magnitude below 1.3% of the codelength of left-to-right (Figure 1, Table 3). Left-to-right is significantly best only on chain-of-thought, and an alternative is significantly better only on path-star and, by about 0.1%, on both protein sources; on the remaining ten corpora neither is. On path-star, the one corpus with a large margin, both winning orders—full reversal and `gf44`—read the answer before the edge list (Appendix I).

4.2 TEXT IS MORE SENSITIVE TO LOCAL SHUFFLING THAN REVERSAL

On text, local shuffling incurs a much larger codelength penalty than full reversal. Reversal preserves adjacency while reversing direction; local shuffling preserves the order of blocks but changes both adjacency and direction within each block. Figure 2 expresses the excess codelength of each as a percentage of that of a global shuffle, which we call its *normalized excess*. On text, the normalized

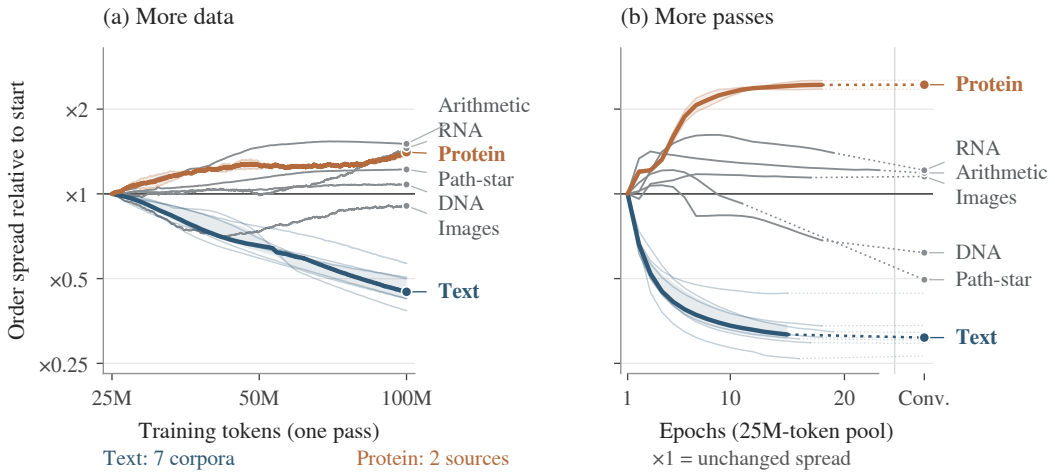

Figure 3: Change of the relative gap D with training, as a multiple of its value at the starting point (log scale); above $\times 1$ the orders drift apart, below $\times 1$ they converge. (a) More data: D read from the moving average of the prequential codelength of the single pass at each N from 25M to 100M, relative to its value at 25M. (b) More passes: D read from the held-out codelength after each epoch over the fixed 25M-token pool, relative to its value after epoch 1; each curve ends at the last epoch that every order of the corpus still reaches in at least three seeds, and the dotted segment joins it to the converged value (rightmost column). Thin lines are individual corpora. Blue: the seven text corpora, with their median (thick line) and interquartile range (band); brown: both protein sources, with their geometric mean and range; gray: the other corpora, labelled individually.

excess of reversal is at most 5.5% in either sign, whereas shuffling within blocks of four already reaches 50–62%. At least half of what left-to-right gains over a global shuffle thus depends on the order within spans of four tokens: for this learner, most of the usable order structure of text is local. Reversal stays far below four-token shuffling on every other collected corpus as well, including the two where its normalized excess is largest in magnitude: 15% against 61% on RNA and -16% against 38% on OMG (Table 4).

4.3 The verdict moves with training

How much order matters, and sometimes which of two orders is cheaper, changes as training proceeds: between 6.25M and 100M tokens, reversal falls behind left-to-right on English Wikipedia and overtakes it on OMG (Section 2.3, Appendix E).

The spread among orders—the difference between their largest and smallest codelengths—changes too. We normalize it as the relative gap

$$D \;=\; \frac{\max_\sigma \ell_\sigma - \min_\sigma \ell_\sigma}{H_{\text{uni}} - \min_\sigma \ell_\sigma},$$

where H_{uni} is the codelength per token using unigram frequencies without context. Here ℓ_σ denotes the moving average $\tilde{\ell}_\sigma(N)$ in the single pass or the held-out codelength per token $\ell_\sigma^{\text{ho}}(e)$ after epoch e under repeated passes. Thus D is the fraction of the distance from the best order to the unigram baseline spanned by the tested orders (Appendix K). Following Su (2026), we separate training longer from seeing more data: from a common base, a single pass over 25M tokens, we extend training either by repeated passes over the same pool until convergence or by new data up to 100M. The two extensions move text and protein in opposite directions (Figure 3; every order on English and both protein sources in Figure 4). On all seven text corpora both narrow the gap, as orders that start behind partly catch up: repeated passes lower D from 0.49–0.90 to 0.21–0.26. On both protein sources both widen it, as differences between orders emerge only as the model learns more: repeated passes raise D from 0.21–0.22 to 0.51–0.53. The other corpora do not share one direction (Appendix K).

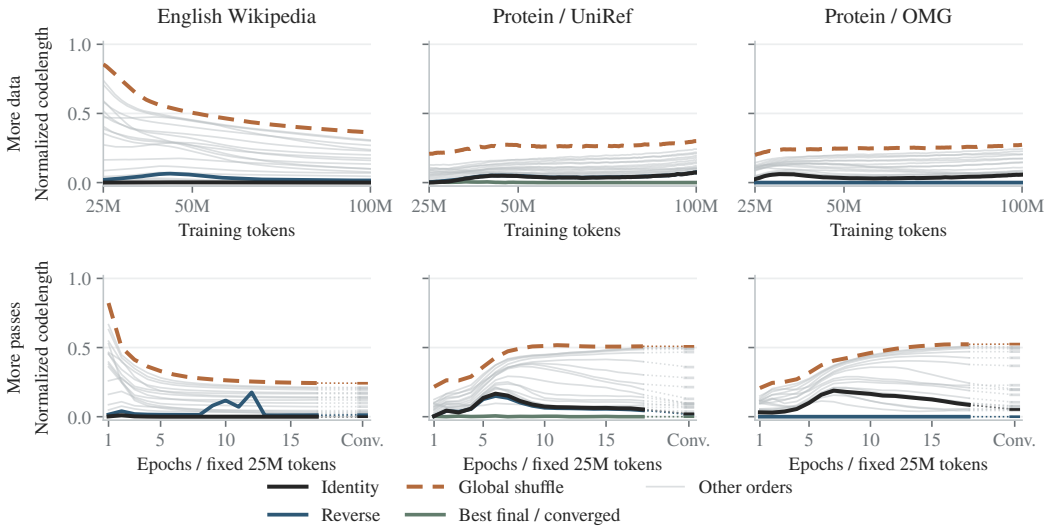

Figure 4: Normalized codelength of every order on English Wikipedia and both protein sources: $(\ell_\sigma - a)/(H_{\text{uni}} - a)$ with $a = \min_\sigma \ell_\sigma$ at each budget or epoch, so that the best order lies at 0, the unigram level at 1, and the highest curve gives the relative gap D. Top: more data, the moving average of the prequential codelength from 25M to 100M. Bottom: more passes, the held-out codelength after each epoch over the fixed 25M-token pool, with dotted connectors to each order's converged value (Conv.). Identity is black, reverse blue, the global shuffle brown dashed and the best order at 100M or at convergence green; other orders are gray.

One reading of this contrast is that it reflects the stage of learning rather than the type of data. At 100M the learner has extracted 42–75% of the unigram entropy on text but about 4% on protein, so protein may still be where text was early in training; arithmetic, where the gap widens although 86% is extracted, shows that this cannot be the whole account (Appendix E). Either way, a comparison of orders at a single budget can mislead in either direction, which is why we compare orders over a range of budgets (Section 2.3).

4.4 Content-dependent orders lower prediction loss

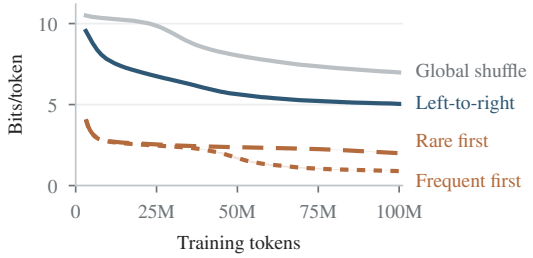

Figure 5: Frequency orders on English Wikipedia: moving average of prediction loss over eight replicates. Dashed curves use content-dependent orders and omit the cost of communicating the order. These runs are separate from the main experiment; its global shuffle is shown for scale.

Two further orders on English Wikipedia sort each window by the global frequency of its tokens, most frequent first (`fqd`) or rarest first (`fqa`). Their average prediction losses over 0.25M–100M are 1.79 and 2.43 bits/token, respectively, compared with 6.05 for left-to-right, the best tested fixed order (Figure 5). Here the order itself provides a content-derived signal about which positions hold more frequent tokens, making the prediction problem easier for the learner. These losses exclude the cost of communicating the order and therefore are not standalone codelengths. The results motivate learning to choose prediction positions from frequency ranks estimated using the revealed context. Whether such a policy can retain the observed gains without access to unseen tokens remains open.

5 Related work

Direction asymmetry. Papadopoulos et al. (2024) report forward–backward perplexity gaps across languages. Luo et al. (2025) establish factorization invariance under the true distribution, identify protocol errors in earlier comparisons, and trace remaining discrepancies, amplified for permuted orders, to positional and locality biases of self-attention. We hold these biases fixed as part of the learner and compare orders throughout training, when their relative performance can change (Sections 3.3 and 4.3).

Generation order and any-order models. The study of order sensitivity in sequence models dates back at least to Vinyals et al. (2016) and Ford et al. (2018). Permutation and any-order models (Uria et al., 2014; Yang et al., 2019; Shih et al., 2022; Hoogeboom et al., 2022; Pannatier et al., 2024), discrete and masked diffusion models (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024) and order-aware image generation (Tian et al., 2024; Pang et al., 2025) all treat order as a design choice made for generation quality, representation learning, or the flexibility and cost of inference. Bachmann & Nagarajan (2024) show that next-token prediction can fail on tasks such as path-star graphs, and Kim et al. (2025) study how masked diffusion orders its tokens. The reversal curse (Berglund et al., 2024), recast by Kitouni et al. (2024) as a failure to learn one joint distribution under different factorizations, shows that a left-to-right learner can fail to acquire a relation stated in one direction and queried in the other, and Prabhudesai et al. (2025) find that masked diffusion makes better use of repeated data than left-to-right training. These studies compare training objectives; we hold the objective and the architecture fixed and use order as a probe of how learnable the data are for a given learner.

Prequential MDL. Prequential codelength provides a practical description length for deep models (Blier & Ollivier, 2018); Bornschein et al. (2023) refine its estimation with replay and forward calibration. Applications include probing (Voita & Titov, 2020) and few-shot model selection (Perez et al., 2021), neither varying the factorization order.

Causal and information-theoretic grounding. The algorithmic Markov condition (Janzing & Schölkopf, 2010) links simple conditionals to the true causal factorization, and adaptation speed has been used as a causal signal (Bengio et al., 2020), with its limits delimited by Le Priol et al. (2021); we measure learning efficiency in the same spirit but make no claim about the process that generated the data. $\mathcal{V}$-information (Xu et al., 2020) is defined relative to a predictive family, can be asymmetric, and can be estimated from finite samples when the family has bounded complexity; our quantity is defined, analogously, relative to a fixed learner.

6 Limitations

Our results characterize order-learnability for one learner (17.1M parameters at $V = 8192$, 1024-token context) trained on 100M tokens; held-out codelengths at convergence are measured only for a fixed 25M-token pool, and the spread between orders can move in either direction with further training, so the results do not establish asymptotic differences or behavior at larger scale. The comparison with the best alternative holds only within our basis of twenty-one fixed orders; calibration covers first-order dependence only; each order runs at its own selected learning rate, which on protein and RNA cannot yet be separated from the effect of the order; and most data types enter through a single corpus. Appendix M discusses each point.

7 Conclusion

Prequential codelength measures how efficiently a fixed learner acquires data under different prediction orders. For the tested learner, left-to-right is significantly best only on chain-of-thought traces; text is more sensitive to broken adjacency than reversed direction; and further training narrows the relative gap between orders on text but widens it on protein. These findings characterize order preference jointly through the data, learner and training budget. The frequency-order results motivate learning prediction schedules from available context. Generalization to larger models and training over variable orders remains open.

AI USE STATEMENT

We used generative AI tools to polish the writing, draft parts of the text, search for related work, and assist in writing code. The authors reviewed all AI-assisted text and code and take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

The estimator is defined in Section 2.2, every order in Table 5, and the model, optimizer, learning-rate selection and training protocol in Section 3.3 and Appendix B. The calibration anchors that validate the instrument are specified in Section 3.4 and their results listed in Table 2. The corpora, their vocabularies and document definitions are given in Section 3.1 and Appendix B; the two generated corpora are specified in Section 3.1 and Appendix I, including a complete example instance. The convergence protocol, including its stopping rule, is given in Appendix K.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. arXiv:2107.03006.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024. arXiv:2403.06963.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1901.10912.

Roger L. Berger. Multiparameter hypothesis testing and acceptance sampling. *Technometrics*, 24 (4):295–300, 1982.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.12288.

Léonard Blier and Yann Ollivier. The description length of deep learning models. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 2220–2230, 2018. arXiv:1802.07044.

Jorg Bornschein, Yazhe Li, and Marcus Hutter. Sequential learning of neural networks for prequential MDL. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.07931.

Andre Cornman, Jacob West-Roberts, Antonio Pedro Camargo, Simon Roux, Martin Beracochea, Milot Mirdita, Sergey Ovchinnikov, and Yunha Hwang. The OMG dataset: An open MetaGenomic corpus for mixed-modality genomic language modeling. In *International Conference on Learning Representations*, 2025. bioRxiv 10.1101/2024.08.14.607850.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George E. Dahl. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2942–2946, 2018. arXiv:1808.07910.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Masaki Kawamura, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Oi, Taishi Nakamura, Takumi Okamoto, Shigeki Ishida, Kakeru Hattori, Youmi Ma, Hiroya Takamura, Rio Yokota, Jun Sakuma, and Naoaki Okazaki. Rewriting pre-training data boosts LLM performance in math and code. In *International Conference on Learning Representations*, 2026. arXiv:2505.02881.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2110.02037.

Dominik Janzing and Bernhard Schölkopf. Causal inference using the algorithmic Markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010. doi: 10.1109/TIT.2010.2060095.

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A. Feder Cooper, Aviya Skowron, John Kirchenbauer, Shayne Longpre, Lintang Sutawika, Alon Albalak, Zhenlin Xu, Guilherme Penedo, Loubna Ben Allal, Elie Bakouch, John David Pressman, Honglu Fan, Dashiell Stander, Guangyu Song, Aaron Gokaslan, Tom Goldstein, Brian R. Bartoldson, Bhavya Kailkhura, and Tyler Murray. The Common Pile v0.1: An 8TB dataset of public domain and openly licensed text. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025. arXiv:2506.05209.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30749–30768, 2025. arXiv:2502.06768.

Ouail Kitouni, Niklas Nolte, Diane Bouchacourt, Adina Williams, Mike Rabbat, and Mark Ibrahim. The factorization curse: Which tokens you predict underlie the reversal curse and more. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2406.05183.

Rémi Le Priol, Reza Babanezhad, Yoshua Bengio, and Simon Lacoste-Julien. An analysis of the adaptation speed of causal models. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pp. 775–783, 2021. arXiv:2005.09136.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848, 2024. arXiv:2310.16834.

Xiaoliang Luo, Xinyi Xu, Michael Ramscar, and Bradley C. Love. Probability consistency in large language models: Theoretical foundations meet empirical discrepancies. *arXiv preprint arXiv:2505.08739*, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T. Freeman, and Yu-Xiong Wang. RandAR: Decoder-only autoregressive visual generation in random orders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 45–55, 2025. arXiv:2412.01827.

Arnaud Pannatier, Evann Courdier, and François Fleuret. σ-GPTs: A new approach to autoregressive models. In *Machine Learning and Knowledge Discovery in Databases. Research Track – European Conference, ECML PKDD 2024, Proceedings, Part VII*, volume 14947 of *Lecture Notes in Computer Science*, pp. 143–159. Springer, 2024. doi: 10.1007/978-3-031-70368-3_9. arXiv:2404.09562.

Vassilis Papadopoulos, Jérémie Wenger, and Clément Hongler. Arrows of time for large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39509–39528, 2024. arXiv:2401.17505.

Suraj Patil, William Berman, Robin Rombach, and Patrick von Platen. aMUSEd: An open MUSE reproduction. *arXiv preprint arXiv:2401.01808*, 2024.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 11054–11070, 2021. arXiv:2105.11447.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. arXiv:2507.15857.

RNAcentral Consortium. RNAcentral 2021: secondary structure integration, improved sequence search and new member databases. *Nucleic Acids Research*, 49(D1):D212–D220, 2021. doi: 10.1093/nar/gkaa921.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2406.07524.

Valerie A. Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A. Kitts, Terence D. Murphy, Kim D. Pruitt, Françoise Thibaud-Nissen, Derek Albracht, Robert S. Fulton, Milinn Kremitzki, Vincent Magrini, Chris Markovic, Sean McGrath, Karyn Meltz Steinberg, Kate Auger, William Chow, Joanna Collins, Glenn Harden, Timothy Hubbard, Sarah Pelan, Jared T. Simpson, Glen Threadgold, James Torrance, Jonathan M. Wood, Laura Clarke, Sergey Koren, Matthew Boitano, Paul Peluso, Heng Li, Chen-Shan Chin, Adam M. Phillippy, Richard Durbin, Richard K. Wilson, Paul Flicek, Evan E. Eichler, and Deanna M. Church. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research*, 27(5):849–864, 2017. doi: 10.1101/gr.213611.116.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2406.04329.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. arXiv:2205.13554.

Jianlin Su. Deconstructing scaling laws: A trio of optimization, architecture and data. `https://kexue.fm/archives/11833`, 2026. Blog post, in Chinese.

Baris E. Suzek, Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, Cathy H. Wu, and the UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015. doi: 10.1093/bioinformatics/btu739.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2404.02905.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 467–475, 2014. arXiv:1310.1757.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*, 2016. arXiv:1511.06391.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 183–196, 2020. doi: 10.18653/v1/2020.emnlp-main.14.

Wikimedia Foundation. Wikimedia Downloads. `https://dumps.wikimedia.org`, 2023. Dump of 2023-11-01, as processed in the Hugging Face dataset `wikimedia/wikipedia`.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:2002.10689.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. arXiv:1906.08237.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.14157.

A ENTROPY, USABLE INFORMATION AND THE RESOURCE-BOUNDED RESTATEMENT

This appendix states in full the argument sketched in Sections 1 and 2.2: why the order of factorization cannot matter to the true distribution, why it can matter to a learner, and which quantity the prequential codelength measures.

Order invariance of the true distribution. Let p be a distribution over windows $x = (x_0, \ldots, x_{n-1})$ and $(\sigma(0), \ldots, \sigma(n-1))$ any ordering of their positions. The chain rule gives

$$p(x) = \prod_{t=0}^{n-1} p\big(x_{\sigma(t)} \mid x_{\sigma(0)}, \ldots, x_{\sigma(t-1)}\big)$$

for every σ, so $-\log p(x)$, and hence the entropy $H(p) = \mathbb{E}_p[-\log p(x)]$, is the same under every order; for the exact distribution, perplexity is invariant under forward, backward and arbitrary permutations (Luo et al., 2025). No property of p alone can therefore distinguish one order from another.

The model class depends on the order. A learner does not represent p but a family $\mathcal{Q}_\sigma = \{q_{\theta,\sigma}\}$ of distributions of the form $q_{\theta,\sigma}(x) = \prod_t q_\theta(x_{\sigma(t)} \mid \text{revealed pairs})$, parameterized by one network of fixed size. The same network is asked to represent different conditionals under different orders: under `id` it must predict a token from its prefix, under `rev` from its suffix, under `global0` from a scattered subset. As sets of joint distributions, $\mathcal{Q}_\sigma$ and $\mathcal{Q}_{\sigma'}$ therefore differ in general. For any q, the expected codelength decomposes as

$$\mathbb{E}_p[-\log q(x)] = H(p) + \mathrm{KL}(p \,\|\, q).$$

Every order therefore has an expected codelength of at least $H(p)$, and reaches it only if $p \in \mathcal{Q}_\sigma$ and the learner finds it. Otherwise the lowest codelength any member of the family attains is $H(p) + \inf_{q \in \mathcal{Q}_\sigma} \mathrm{KL}(p \,\|\, q)$, and since the family depends on σ, so does this infimum: two orders can differ even with unlimited data, through how closely one network of fixed size can approximate p under each.

What the prequential codelength accumulates. Before convergence a second difference arises. Let the corpus be presented in batches $b = 1, 2, \ldots$ of documents drawn independently from p, and let $\theta_{<b}$ be the parameters after the first $b-1$ batches. Because $\theta_{<b}$ does not depend on batch b, the decomposition above applies to each batch conditionally on the past, and the expected prequential codelength of Section 2.2 is

$$\mathbb{E}\big[L_\sigma(N)\big] = \sum_b \Big(H(p_b) + \mathbb{E}\,\mathrm{KL}\big(p_b \,\|\, q_{\theta_{<b},\sigma}\big) \Big),$$

where p_b denotes the distribution of one batch. The first term is identical for every order. The second is the accumulated excess of the learner over the true distribution, which depends on σ both through the limit the learner approaches and through how fast it approaches it. This is the expectation of equation 1; when two orders code the same batches, the first term cancels exactly and not only in expectation (Section 2.2). The difference of two orders' prequential codelengths, and hence the excess codelength of Section 2.3, is thus a difference in accumulated redundancy: an estimate of how much less efficiently one order is learned than another by the given learner on the given data.

Usable information. This quantity is defined relative to a learner, not to the distribution. $\mathcal{V}$-information (Xu et al., 2020) makes the analogous move for mutual information: it is defined relative to a predictive family $\mathcal{V}$, can be estimated from finite samples with PAC guarantees when $\mathcal{V}$ has bounded complexity, and, unlike Shannon information, can be asymmetric, so that one variable can be easy to predict from another but not the reverse. Order-learnability is asymmetric in the same sense: a corpus can be more learnable forward than backward for one learner although the two orders carry identical Shannon information. Every quantity in this paper is accordingly stated relative to the learner of Section 3.3.

B CORPORA AND MODEL CONFIGURATION

Table 1 lists the corpora and their settings. The learner of Section 3.3 has $d_{\text{model}} = 384$, six layers, six attention heads and a context of 1024 positions. Its size differs across corpora only through the token embedding and the output layer, which scale with V. A document is a Wikipedia article, a 1024-base genomic block, a single RNA sequence, a single amino-acid sequence of length at least 1024, a complete reasoning trace, a book, an image (1024 VQ codes), a path-star graph, or a packed block of eight long multiplications.

One window per document. A long document yields many 1024-token windows. If several were used, the model would update on one window before coding another from the same document, so each window would be coded partly from what the model has already learned elsewhere in that document, such as its topic, vocabulary, named entities and style. What it has learned depends on the order in which the windows enter the stream rather than on the order under study, which imposes a second, document-level order that it does not control and that can conflict with it. We therefore draw a single window per document, at a random offset.

Table 1: Corpora and settings. *Tokens* is the unit a position holds (BPE: byte-pair encoding; VQ: vector quantization) and V the vocabulary size; *window* is the number of tokens per document window; *orders* is the number of orders run on the corpus; H_{uni} is the unigram entropy under the corpus's own vocabulary, in bits/token; *params* is the number of trainable parameters of the learner on the corpus, including the token embedding and the output layer. Every corpus provides a training pool of 10^8 tokens, one window per document. Besides the 21 orders of Section 3.2, the image corpus adds four two-dimensional orders (snake, pyramid, Hilbert and column-major traversals of the 32×32 grid) and the graph corpus two per-problem orders (tail reversal and goal-first on the answer); see Appendix I. The graph window is one complete instance: 55 edges (220 tokens), the query (4) and the 11-edge answer path (44). Instances long enough to fill 1024 tokens are too hard for the model to learn under any order. Sources: Wikipedia (Wikimedia Foundation, 2023), pre-1929 books from the Common Pile (Kandpal et al., 2025), SwallowCode-v2, the second release of the corpus of Fujii et al. (2026), OpenMathReasoning (Moshkov et al., 2025), GRCh38 (Schneider et al., 2017), RNAcentral (RNAcentral Consortium, 2021), UniRef50 (Suzek et al., 2015), OMG_prot50 (Cornman et al., 2025), ImageNet (Deng et al., 2009) tokenized with the aMUSEd VQ codebook (Patil et al., 2024), and path-star graphs after Bachmann & Nagarajan (2024).

corpus	source	tokens	V	window	orders	H_{uni}	params
language / en	Wikipedia (en)	BPE	8192	1024	21	10.392	17.1M
language / ja	Wikipedia (ja)	BPE	8192	1024	21	10.873	17.1M
language / ru	Wikipedia (ru)	BPE	8192	1024	21	10.828	17.1M
language / zh	Wikipedia (zh)	BPE	8192	1024	21	11.033	17.1M
books	pre-1929 books	BPE	8192	1024	21	10.102	17.1M
code	SwallowCode-v2 (Python)	BPE	8192	1024	21	9.914	17.1M
chain-of-thought	OpenMathReasoning	BPE	8192	1024	21	8.662	17.1M
DNA	GRCh38 (human)	character	5	1024	21	1.976	10.8M
RNA	RNAcentral	character	5	1024	21	2.006	10.8M
protein / UniRef	UniRef50	character	21	1024	21	4.158	10.8M
protein / OMG	OMG_prot50	character	21	1024	21	4.152	10.8M
arithmetic	generated (8×8 digits)	character	15	1024	21	3.619	10.8M
images (VQ)	ImageNet-21k	VQ codebook	8192	1024	25	11.762	17.1M
graph	generated (path-star)	label vocabulary	1004	268	23	6.539	11.6M

C CALIBRATION ANCHORS

Each chain anchor of Section 3.4 is a left-to-right Markov chain over $V = 64$ symbols: with probability 0.95 a token is a fixed function of its predecessor, and otherwise it is drawn uniformly from all 64 symbols, the predicted one included. Given its predecessor, a token therefore takes the predicted value with probability $p = 0.95 + 0.05/64$ and each of the other 63 values with probability $q = 0.05/64$, so its conditional entropy is

$$H \;=\; -p\log_2 p \;-\; 63\,q\log_2 q \;=\; 0.0692 + 0.5080 \;=\; 0.5773 \text{ bits.}$$

The chain is then written into the window along σ^*. Because the chain rule factorizes the distribution of the window in every order, the entropy of the window is the same under every order, and equals 0.5773 bits for every token after the first; only σ^* presents it as a sequence of one-step conditionals. Table 2 gives the result for each anchor.

Table 2: Calibration anchors. Each chain anchor writes a lag-1 Markov chain along a target order σ^*, so that every order has the same entropy (0.5773 bits for every token after the first) and σ^* is the easiest order to learn by construction; the iid anchor has no order structure and entropy 6 bits/token under every order. $\bar{\ell}$ is the average codelength $\bar{\ell}_\sigma$ over 0.25M–100M (Section 2.3; bits/token, mean of eight replicates) under ten orders: identity and the nine targets listed in Section 3.4. The runner-up is the order with the second-lowest $\bar{\ell}$; the margin is $\bar{\ell}$(runner-up) $-$ $\bar{\ell}(\sigma^*)$ with its paired 95% t-interval, and an anchor passes when σ^* is the argmin and the interval lies above zero. For the iid anchor, the last two columns give the spread of $\bar{\ell}$ across all ten orders and the widest interval of any order against identity; every such interval contains zero.

anchor	σ^*	$\bar{\ell}(\sigma^*)$	runner-up	$\bar{\ell}$(runner-up)	margin [95% CI]	pass
chain	`global0`	0.5846	`eo`	2.1104	$1.5258\ [1.4930, 1.5586]$	yes
chain	`ls64`	0.5845	`eo`	1.0567	$0.4722\ [0.4626, 0.4819]$	yes
chain	`rr8`	0.5848	`gf32`	0.8995	$0.3147\ [0.3043, 0.3250]$	yes
chain	`skd64`	0.5844	`eo`	0.7623	$0.1779\ [0.1732, 0.1826]$	yes
chain	`sk4`	0.5845	`id`	0.7608	$0.1763\ [0.1740, 0.1786]$	yes
chain	`eo`	0.5839	`id`	0.7267	$0.1428\ [0.1404, 0.1451]$	yes
chain	`fim50`	0.5843	`id`	0.5894	$0.0051\ [0.0050, 0.0053]$	yes
chain	`gf32`	0.5844	`id`	0.5890	$0.0046\ [0.0045, 0.0046]$	yes
chain	`rev`	0.5845	`id`	0.5858	$0.0013\ [0.0013, 0.0013]$	yes
iid	—	$\bar{\ell} \in [6.001416, 6.001429]$ over all ten orders; spread 1.4×10^{-5}, widest CI 7.4×10^{-5}				yes

D MARGIN OF LEFT-TO-RIGHT IN BITS/TOKEN

Table 3 gives the margins of Figure 1 in bits/token, with the per-contrast and the Bonferroni-adjusted intervals of Section 4.1 and the rank of identity among all orders of each corpus.

E PROTEIN

This appendix collects the analyses specific to protein, whose two sources are UniRef and OMG (Section 3.1).

A second source. Because protein alone runs against the narrowing seen on text (Section 4.3), it is replicated on a second source, OMG, whose proteins come from environmental metagenomes rather than sequenced organisms. The two sources agree in the direction of the change on both axes (Figure 3), so the widening is not specific to one corpus.

Direction. The two sources disagree on the direction of reading. On UniRef, reading backward has a higher codelength than reading forward early in training and no higher by its end: the smoothed codelength difference $\tilde{\ell}_{\mathrm{rev}}(N) - \tilde{\ell}_{\mathrm{id}}(N)$ is $+0.0037$ ($[+0.0029, +0.0045]$) bits/token at 6.25M and $+0.0000$ ($[-0.0003, +0.0004]$) at 100M. On OMG, it falls from $+0.0036$ ($[+0.0029, +0.0042]$) at 6.25M to -0.0096 ($[-0.0102, -0.0091]$) at 100M. Full reversal is therefore the best alternative on OMG, with a normalized excess of -16% (Table 4); it is cheaper than identity by 0.116% of the codelength of identity, and the margin survives the correction for selecting among alternatives (Figure 1, Table 3). On UniRef the best alternative is `fim75`, whose margin survives it as well. The two sources thus agree in the sign and the significance of the margin but not in the order that attains it. We have not determined why the direction differs between them.

Interpretation. The corpora on which the gap narrows and those on which it widens also differ in how far learning has progressed. At 100M, identity has extracted 42–75% of the unigram entropy on

Table 3: Margin of left-to-right over its best alternative among all orders of the corpus. The best alternative is the order whose excess codelength over 0.25M–100M is smallest; the margin is that excess codelength (bits/token), so a positive margin means every alternative has a higher codelength than identity, with its paired 95% t-interval over eight replicates. *Margin / codelength* divides the margin by $\bar{\ell}_{\mathrm{id}}$, the average codelength of identity over the same range and orders the rows. Rank is the position of identity among all orders of the corpus (1 = lowest $\bar{\ell}_\sigma$); order counts are given in Table 1. *Bonf. 95% CI* is the Bonferroni-adjusted simultaneous interval over the $k-1$ alternatives of a corpus with k orders, the interval to read for a negative margin (Section 4.1); it is not needed for a positive margin, which is tested by the intersection–union principle.

corpus	best alternative	margin	95% CI	Bonf. 95% CI	margin / codelength	rank
chain-of-thought	`fim75`	$+0.0249$	$[+0.0183, +0.0315]$	—	$+0.767\%$	1
code	`gf32`	$+0.0032$	$[-0.0278, +0.0343]$	—	$+0.093\%$	1
English	`gf8`	$+0.0013$	$[-0.0160, +0.0186]$	—	$+0.022\%$	1
RNA	`rr8`	$+0.0001$	$[-0.0033, +0.0035]$	—	$+0.004\%$	1
books	`rev`	-0.0006	$[-0.0214, +0.0203]$	$[-0.0411, +0.0399]$	-0.009%	2
images (VQ)	`rev`	-0.0029	$[-0.0090, +0.0032]$	$[-0.0152, +0.0093]$	-0.027%	2
DNA	`fim50`	-0.0012	$[-0.0030, +0.0006]$	$[-0.0047, +0.0023]$	-0.064%	2
protein (UniRef)	`fim75`	-0.0041	$[-0.0050, -0.0032]$	$[-0.0059, -0.0023]$	-0.102%	3
protein (OMG)	`rev`	-0.0047	$[-0.0049, -0.0044]$	$[-0.0052, -0.0041]$	-0.116%	2
Chinese	`gf32`	-0.0117	$[-0.0433, +0.0199]$	$[-0.0731, +0.0496]$	-0.173%	3
Japanese	`rev`	-0.0205	$[-0.0614, +0.0203]$	$[-0.0999, +0.0588]$	-0.330%	2
Russian	`rev`	-0.0599	$[-0.1121, -0.0078]$	$[-0.1613, +0.0414]$	-1.029%	2
arithmetic	`fim50`	-0.0162	$[-0.1336, +0.1011]$	$[-0.2442, +0.2118]$	-1.298%	2
graph	`rev`	-0.3534	$[-0.4075, -0.2993]$	$[-0.4603, -0.2465]$	-8.686%	12

text but only about 4% on protein (Table 6), so the two may lie at different points of one trajectory, along which the gap first opens as structure is learned and later closes as slower orders catch up. Arithmetic shows that this cannot be the whole account: identity extracts 86% of the unigram entropy there, more than on code and chain-of-thought, yet the gap widens under both extensions. The present runs cannot separate the stage of learning from the type of data, because no protein corpus is observed late in learning, and early in learning D on text is pinned near 1 and therefore uninformative. The fraction of H_{uni} is, moreover, only a proxy for progress, since the entropy of each corpus is unknown. One explanation of the opposite directions on text and protein is that the text gap reflects mainly the speed of learning, which slower orders partly recover with further passes, whereas the protein gap reflects mainly the level an order can attain, which emerges only with sufficient training. We have not tested this mechanism.

F Reversal and local shuffles on every corpus

Table 4 gives the normalized excess (Section 4.2) of full reversal and of every block width of $\mathtt{ls}\langle w\rangle$, the quantity plotted for `rev`, `ls4` and `ls64` in Figure 2 (Section 4.2).

Generated corpora. Neither generated corpus follows this pattern: on arithmetic, full reversal has a higher normalized excess than `ls4`; on the graphs, it has a lower codelength than identity (Appendix I), and $\mathtt{ls}\langle w\rangle$ is not monotone in the block width.

The four Wikipedia editions. The editions rank the orders almost identically, with pairwise Spearman correlations of 0.97–0.99 across the nineteen orders other than identity and the global shuffle. Russian stands apart in two respects. It is the only edition on which full reversal is significantly cheaper than left-to-right, though only before the adjustment of Section 4.1 (a normalized excess of -2.0%, against $+4.7\%$ on English), and it is the least sensitive to local shuffling, 7–11 points below the next edition at block widths from 16 to 64. Both are consistent with two features of Russian: its case system leaves word order comparatively free, so position carries less information, and an attributive adjective precedes its noun but takes its ending from it, so a reversed reading sees the noun before the ending it determines. We have not tested either explanation.

Table 4: Normalized excess of full reversal and of local shuffles of increasing block width: the excess codelength over 0.25M–100M as a percentage of that of a global shuffle on the same corpus, so that 0 means the codelength of identity and 100 that of a global shuffle. The ratio is dimensionless and therefore comparable across corpora. Rows above the rule are the eleven collected corpora, with protein shown for both sources; the two generated corpora below it do not follow the pattern of the others.

corpus	rev	ls4	ls8	ls16	ls32	ls64
English	4.7	55.2	68.2	79.3	87.9	89.5
Russian	−2.0	52.7	64.3	71.7	76.9	78.9
Chinese	0.9	61.5	71.4	79.1	85.2	96.1
Japanese	−0.9	59.9	84.8	85.9	90.5	93.2
books	0.0	51.6	60.1	66.8	71.7	86.5
code	2.2	50.2	57.8	67.7	74.6	76.1
chain-of-thought	5.5	51.0	68.8	78.8	83.6	87.0
DNA	0.3	55.7	70.3	74.9	79.0	72.9
RNA	15.3	60.8	69.2	70.5	76.5	80.5
protein (UniRef)	1.9	23.8	32.2	41.9	51.1	60.3
protein (OMG)	−16.2	38.1	48.8	60.2	69.5	77.7
images (VQ)	−1.1	37.3	42.6	50.9	53.9	69.2
arithmetic	68.0	52.2	57.8	64.9	74.8	77.3
graph	−38.2	38.0	25.1	3.7	100.3	100.0

G THE ORDERS

We consider twenty-one orders drawn from ten families, each of which probes a single axis. The identity order (`id`) reads left to right and serves as the reference for every contrast. Reverse (`rev`) reads from right to left and probes direction. Local shuffle ($\texttt{ls}\langle w\rangle$, for $w \in \{4, 8, 16, 32, 64\}$) permutes positions randomly within blocks of width w and thus provides a graded control of locality. Skeleton infill ($\texttt{sk}\langle b\rangle$) first places every b-th position and then fills in the remainder left to right, probing coarse-to-fine generation with a single coarse level. Dyadic refinement (`skd64`) is its recursive variant and makes the refinement hierarchical: it places every 64th position, then the midpoints between them, then the midpoints of those, halving the stride at each level until every position is filled. Evens-then-odds (`eo`) coincides with $\texttt{sk}\langle b\rangle$ at $b = 2$; we count it as a family of its own because it is the only skeleton order whose skeleton is as large as its infill. Fill-in-middle ($\texttt{fim}\langle p\rangle$) reads the prefix and suffix before the middle span, which occupies $p\%$ of the window. Goal-first ($\texttt{gf}\langle g\rangle$) moves the last g positions to the front, which directs information flow backward without inverting any conditional within a segment. Round-robin ($\texttt{rr}\langle k\rangle$) partitions the document into k contiguous segments and reads one token from each in turn. Global shuffle (`global0`) is a uniformly random order. Like every order of the main experiment, it is a single permutation, drawn once and applied to every window; it is therefore not the per-window random order of any-order training in σ-GPT or masked diffusion, whose effect on learning this study does not measure. The image and path-star corpora add orders of their own, which are not among the twenty-one. On the path-star task, whose window ends with a 44-token answer of eleven four-token edges (Appendix I), `gf44` is goal-first with the whole answer moved to the front, and tail reversal (`tr44`) reads the answer backward edge by edge, each edge left to right; the four two-dimensional image orders are defined in Appendix I. Figure 6 draws each family on a 16-token window, and Table 5 gives the precise definition of every order.

Positions of a window are $0, \ldots, n-1$, with $n = 1024$ except on the graph corpus ($n = 268$). An order lists these positions as $(\sigma(0), \ldots, \sigma(n-1))$, with prediction steps numbered $0, \ldots, n-1$; the model predicts $x_{\sigma(0)}$ first. Except for the two content-dependent orders of Section 4.4, every order is a single permutation, fixed before training and shared by every window of the corpus; the random orders are drawn once from a seeded generator.

H STRUCTURE EXTRACTED BY EACH CORPUS

Table 6 gives, for each corpus, the structure that identity extracts beyond unigram frequencies at 100M.

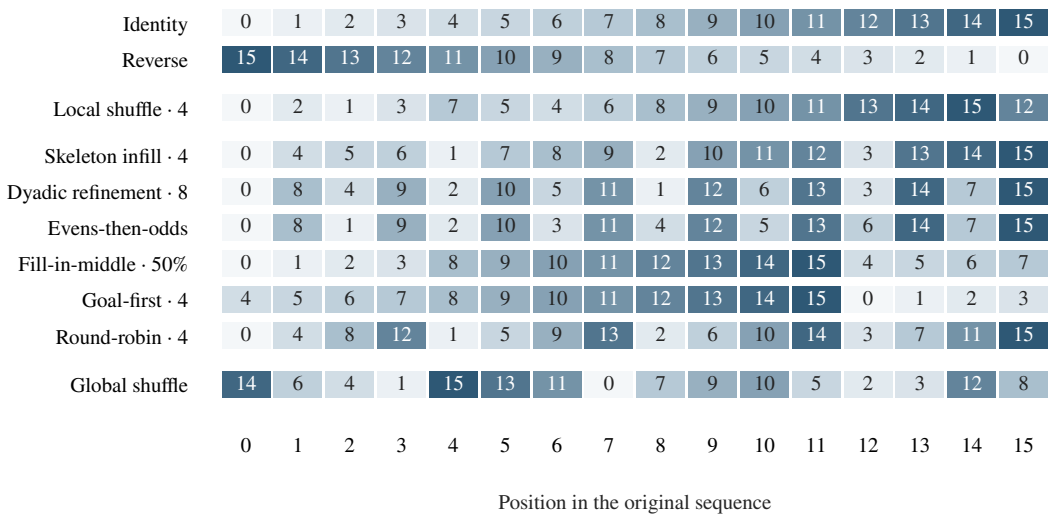

Figure 6: The order families on a 16-token window. Columns are positions of the original sequence, rows are orders, and each cell gives the step at which that position is predicted (0 = first; darker = later); reading a row in increasing cell value traces $\sigma(0), \ldots, \sigma(n-1)$. Parameters are scaled to $n = 16$, so the dyadic order starts at stride 8 rather than 64. `ls⟨4⟩` and `global0` are single permutations drawn once (seed 0 here) and shared by every window.

I THE GRAPH CORPUS AND THE TWO-DIMENSIONAL ORDERS

Path-star problems. Each document of the graph corpus is one path-star search problem. Fifty-six labels are drawn without replacement from 1000 as the nodes of a star graph: a start node from which five disjoint paths (arms) emanate, each of eleven edges and eleven further nodes, 55 edges in total. A document has three parts: (1) all 55 edges, each written `source,target|`, in random order, the last terminated by `/`; (2) the query `start,goal=`, where the goal is the end node of a randomly chosen arm; and (3) the answer, the eleven edges from the start to the goal in path order, each written `source,target|`. Every label and every separator is one token, so a problem has exactly $55 \times 4 + 4 + 11 \times 4 = 268$ tokens and the window is one complete problem. To generate the answer left to right, the model must know which arm leads to the goal before it writes the first edge, which requires searching outward along the arms; read backward from the goal, every node has in-degree one, so each step requires a single lookup of its unique predecessor in the edge list. This backward reading is not what makes an order cheap: tail reversal (`tr44`), which reads the edge list first and then the answer from the goal back, is no cheaper than left-to-right (excess codelength -0.0004, $[-0.0164, +0.0156]$), whereas the two orders that are, full reversal (-0.3534) and `gf44` (-0.3462), both place the answer before the edge list (Table 3).

The first problem of the training set is shown below (start 948, goal 522; Figure 7); the edge list is broken after every eight edges for typesetting and contains no line breaks in the data.

```
803,210|829,276|276,044|044,460|518,032|212,827|320,015|917,832|
948,619|338,212|142,739|948,410|092,271|836,934|969,083|948,623|
015,335|135,092|379,624|491,338|315,312|410,969|624,433|532,142|
619,236|210,475|310,183|312,532|475,491|353,967|271,287|287,353|
460,274|601,803|546,601|855,101|739,379|366,011|083,315|623,001|
101,898|001,865|967,836|832,522|236,546|948,635|635,855|274,917|
685,310|183,366|948,320|032,685|335,135|865,829|898,518/
948,522=
948,623|623,001|001,865|865,829|829,276|276,044|
044,460|460,274|274,917|917,832|832,522|
```

Two-dimensional orders. The image corpus tests four additional two-dimensional orders. Each image is encoded as a 32×32 grid of codebook indices and flattened row-major into 1024 tokens, so `id` is the raster scan. Snake scans the rows but reverses every other row, so that consecutive steps are

Table 5: The orders. The first 21 rows are run on every corpus; the image and graph corpora add the orders in the last two groups. *Blocks*, *segments* and *strides* are in positions.

order	definition (reading order)
`id`	$0, 1, \ldots, n-1$.
`rev`	$n-1, n-2, \ldots, 0$.
`ls`$\langle w \rangle$, $w \in \{4, 8, 16, 32, 64\}$	Positions are cut into consecutive blocks of w; blocks are read in order, positions within each block in a random permutation (a different one per block).
`sk`$\langle b \rangle$, $b \in \{4, 8, 16, 32, 64\}$	The skeleton $0, b, 2b, \ldots$ left to right, then all remaining positions left to right.
`eo`	$0, 2, 4, \ldots$, then $1, 3, 5, \ldots$; identical to `sk`$\langle b \rangle$ with $b = 2$.
`skd64`	The stride-64 skeleton $0, 64, 128, \ldots$; then the odd multiples of 32, of 16, …, of 1, each level left to right, halving the stride until every position is placed.
`fim`$\langle p \rangle$, $p \in \{50, 75\}$	A centered middle span of $m = \lfloor pn/100 \rfloor$ positions is deferred: the prefix before it and the suffix after it are read left to right, then the middle span left to right.
`gf`$\langle g \rangle$, $g \in \{8, 32\}$	The last g positions left to right, then positions $0, \ldots, n-g-1$ left to right.
`rr`$\langle k \rangle$, $k \in \{4, 8\}$	The window is cut into k contiguous segments of equal length (the last one shorter when k does not divide n); one position is read from each segment in turn, left to right within each segment.
`global0`	A uniformly random permutation of all n positions.
image corpus (32×32 grid, flattened row-major; `id` is the raster scan)	
snake	Rows top to bottom, alternate rows right to left, so that consecutive steps are grid neighbors.
column-major	Columns left to right, each top to bottom.
Hilbert	The grid along a Hilbert curve.
pyramid	The stride-16 subgrid, then the not-yet-read positions of the stride-8, 4, 2 and 1 subgrids, each level in raster order.
graph corpus (the last 44 positions are the answer: eleven edges of four tokens)	
`gf44`	`gf`$\langle g \rangle$ with $g = 44$: the whole answer first, then the edge list and the query.
`tr44`	Edge list and query left to right, then the eleven answer edges from the last to the first, each edge's four tokens left to right.

Table 6: Structure extracted by identity at $N = 100$M. $\tilde{\ell}_{\mathrm{id}}$ is the moving average of the prequential codelength of identity at 100M (Section 2.3) and $S = H_{\mathrm{uni}} - \tilde{\ell}_{\mathrm{id}}$ the structure it extracts beyond unigram frequencies, both in bits/token. S spans nearly two orders of magnitude across corpora, which is why cross-corpus comparisons in Section 4 use dimensionless ratios.

corpus	H_{uni}	$\tilde{\ell}_{\mathrm{id}}$ at 100M	S
language / en	10.392	5.043	5.349
language / ja	10.873	5.149	5.724
language / ru	10.828	4.756	6.072
language / zh	11.033	5.584	5.449
books	10.102	5.817	4.285
code	9.914	2.519	7.395
chain-of-thought	8.662	2.484	6.179
DNA	1.976	1.791	0.185
RNA	2.006	1.897	0.109
protein / UniRef	4.158	4.000	0.158
protein / OMG	4.152	3.992	0.159
arithmetic	3.619	0.513	3.106
images (VQ)	11.762	10.377	1.385
graph	6.539	3.633	2.905

always grid neighbors; column-major scans the columns; Hilbert traverses the grid along a Hilbert curve, which preserves two-dimensional locality; pyramid first reads the stride-16 subgrid and then



Figure 7: The star graph of the problem above. The start node (948) is black, the arm that leads to the goal (522) is brown, and the other four arms are gray; each arm has eleven edges. The answer lists the brown arm from the start outward.

the not-yet-read positions of the stride-8, 4, 2 and 1 subgrids, so that each level is a coarser image. Figure 8 draws the five orders on an 8×8 grid. None of the four has a lower codelength than the raster scan, not even snake and Hilbert, whose consecutive steps always fall on neighboring cells: identity ranks second of twenty-five orders, behind only full reversal (Table 3).

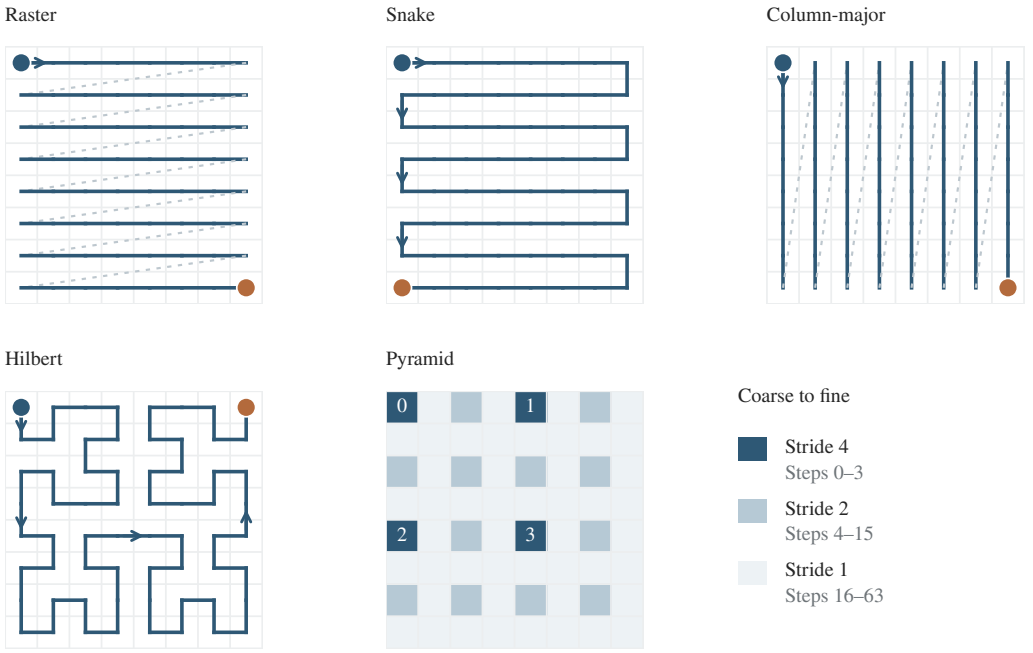

Figure 8: Reading order of the five image orders on an 8×8 grid. For raster, snake, column-major and Hilbert, the solid line joins consecutive steps from the blue dot (first) to the brown dot (last), arrows mark the direction, and dashed segments are jumps between non-adjacent cells. Pyramid is shaded by refinement stage: stride 4 (steps 0–3, numbered), stride 2 (steps 4–15) and stride 1 (steps 16–63). The experiments use the 32×32 grid, on which pyramid starts at stride 16 rather than 4.

J Learning rates

Table 7 gives the learning rate selected for each corpus and order.

K Spread among orders, order by order

Figures 9 and 10 show every order behind the spreads of Section 4.3: in the single pass, and in the convergence experiment.

Table 7: Learning rate selected for each (corpus, order) pair, in units of 10^{-4}. Each rate is chosen on the learning-rate tuning split, disjoint from the training and evaluation pools, from powers of two times 3×10^{-4} (Section 3.3), and held fixed across replicates. Blank cells are orders not run on that corpus.

order	en	ja	ru	zh	books	code	CoT	DNA	RNA	UniRef	OMG	arith.	images	graph
id	12	12	12	12	12	12	12	3	1.5	1.5	1.5	6	12	1.5
rev	12	12	12	12	12	12	12	3	0.75	1.5	3	6	12	3
ls4	12	12	12	12	12	12	12	3	1.5	1.5	1.5	6	12	3
ls8	12	24	12	12	12	12	24	3	0.75	1.5	1.5	6	12	1.5
ls16	12	12	12	12	12	12	12	3	1.5	1.5	1.5	6	12	3
ls32	12	12	12	12	12	12	12	3	0.75	1.5	1.5	6	12	12
ls64	12	12	12	6	6	12	12	3	0.75	1.5	1.5	6	12	6
sk4	6	12	6	6	12	24	24	3	1.5	1.5	3	3	12	3
sk8	12	12	12	12	12	12	12	1.5	0.75	1.5	1.5	6	12	6
sk16	12	12	12	12	12	12	12	3	1.5	1.5	3	6	12	1.5
sk32	12	12	12	6	12	12	12	3	0.75	1.5	1.5	6	12	3
sk64	12	12	12	6	12	12	12	3	1.5	1.5	1.5	6	12	3
eo	12	12	12	12	12	12	24	3	0.75	3	1.5	6	12	6
skd64	12	12	12	12	6	12	12	1.5	0.375	1.5	1.5	3	12	3
fim50	12	12	12	12	12	12	12	6	1.5	1.5	1.5	6	12	3
fim75	12	12	12	12	12	12	12	3	0.75	3	1.5	6	12	1.5
gf8	12	12	12	12	12	24	12	3	1.5	1.5	1.5	6	12	3
gf32	12	12	12	12	12	24	12	1.5	0.75	3	1.5	6	12	3
rr4	12	12	12	12	12	12	12	3	0.75	3	3	3	12	3
rr8	12	12	12	12	12	12	12	6	3	3	3	6	12	3
global0	12	12	6	12	6	6	12	0.75	0.75	0.75	1.5	6	12	6
snake													12	
column-major													12	
Hilbert													12	
pyramid													12	
gf44														3
tr44														1.5

Convergence experiment. Every order is trained for repeated epochs on a fixed 25M-token pool, at the learning rate and schedule of the prequential pass, with the rate capped at 1.2×10^{-3}, which halves it for seven orders (ls8 on Japanese; sk4, gf8 and gf32 on code; eo, ls8 and sk4 on chain-of-thought); absent decay, a plateau reflects convergence rather than a vanishing step size. The held-out pool is disjoint from the training and tuning pools and holds 10,000 windows (about 10M tokens; 38,500 windows on the graph corpus), except on books, whose source provides only 122,082 books at one window per book and leaves 4,894 windows (5.0M tokens) after the other pools. After each epoch e the held-out pool is scored in two halves, split by document; $\ell_\sigma^{\mathrm{ho}}(e)$ denotes the held-out codelength per token. Training stops once the codelength on the selection half has failed for four consecutive epochs to improve on its best by more than 5×10^{-4} bits/token, or at 64 epochs; the converged value is the codelength on the report half at the epoch that minimizes the selection half, averaged over five seeds. Runs stop after 6–64 epochs (median 21); only three orders on arithmetic (ls8, ls16, rr8) reach the cap, in six of fifteen runs, and none is arithmetic's best or worst order at convergence.

Absolute and relative gap. Besides the relative gap D of Section 4.3, we report the absolute gap $G = \max_\sigma \ell_\sigma - \min_\sigma \ell_\sigma$ in bits/token, which is comparable only within a corpus; because a small denominator can inflate a negligible gap in D, the two are read together. Here ℓ_σ is the moving average $\tilde{\ell}_\sigma(N)$ in the single pass and the held-out codelength $\ell_\sigma^{\mathrm{ho}}(e)$ after each epoch under repeated passes. On text, repeated passes lower G to 0.37–0.54 of its first-epoch value; new data lowers D to 0.39–0.57 of its 25M value, from 0.51–0.91 to 0.26–0.43, and G to 0.52–0.85, as the best order gains 1.1–1.8 bits/token and the worst, a global shuffle on every text corpus, 2.1–3.7. On protein, repeated passes grow G 5.5-fold on UniRef and 5.9-fold on OMG, and new data grows G 1.90- and 1.75-fold and D 1.45- and 1.36-fold. G and D move together on both, and the gap between the orders that are best and worst at the starting point narrows to 0.36–0.54 of its value on text and widens 5.5- to 5.9-fold on protein, so neither result is an artefact of the measure. On DNA, repeated passes grow

G 1.83-fold but shrink D to 0.62 of its first-epoch value, and new data leaves D nearly unchanged (1.08-fold), so the direction is indeterminate. On RNA, arithmetic and images, repeated passes grow G 2.1- to 5.0-fold and D 1.15- to 1.21-fold; new data widens D on arithmetic and RNA (1.51- and 1.45-fold) and narrows it on images (0.91-fold). On the graphs, repeated passes narrow the gap (G to 0.61 of its first-epoch value, D from 0.43 to 0.21), whereas new data widens D 1.22-fold. At 25M every order on every corpus already codes below H_{uni}; at smaller budgets the worst order on several text corpora does not, which would pin D near 1 however the gap evolves.

L CODELENGTH CURVES OF EVERY ORDER

Figure 11 shows the prequential codelength of every order on every corpus.

M LIMITATIONS IN DETAIL

The measurement covers a finite range and has not converged. Every result is read from codelength curves between 0.25M and 100M token. At 100M, the text corpora are still improving, whereas the biological corpora remain within 4–10% of their unigram entropy. Once training converges, the spread between orders can move in either direction (Section 4.3), so the level of the spread at any one budget is not a fixed property of a corpus.

The measure is defined relative to a learner. The order invariance of entropy is a property of the optimal predictor, and a finite learner inherits it only to the extent that it recovers that predictor. Every codelength, ratio and margin is defined relative to a single model (17.1M parameters at $V = 8192$) at a single context length (1024); whether the effects persist at larger scales remains untested; since further training narrows the gaps on text (Section 4.3), we cannot rule out that a larger learner narrows them as well. Order at the scale of the whole document, such as the order of sections or paragraphs, lies beyond what a 1024-token window can probe.

Each order has its own learning rate. Each order's learning rate is selected separately, as the best of a grid spaced by a factor of two on one seed of the learning-rate tuning split (Section 3.3), so an order can gain or lose against identity through the rate it is given as well as through the order itself. Where margins are small this matters: on both protein sources and on RNA, the reversal, goal-first and fill-in-middle orders that differ from identity by more than 0.002 bits/token are exactly those selected at a different rate from identity—twice it for the orders that beat identity on protein, half of it for full reversal, `fim75` and `gf32` on RNA (Table 7)—while those selected at identity's rate stay within 0.002 bits/token of it. The protein margins and the excess codelength of reversal on RNA are therefore not yet separated from the selection of the learning rate. The calibration anchors run every order at one common rate (Section 3.4), so they do not test this step.

The order basis is selected rather than exhaustive. The twenty-one orders form ten regular one-parameter families, which cover a vanishing fraction of the $1024!$ possible orders and exclude adaptive orders that depend on the content being read. Every order is, moreover, a fixed permutation (Section 3.3); in particular, the global shuffle is not the per-window random order of any-order training, so the study does not measure the codelength of learning under that objective. The comparison with the best alternative therefore holds only within this basis, and a better order may exist outside it. Monotonicity within a family is in part a consequence of the same design, since each family is parameterized by perturbation strength; for this reason Table 4 reports it corpus by corpus rather than as a general law.

Calibration covers first-order dependence only. All anchors are lag-1 chains (Section 3.4): with a single parent per position at distance one, the excess codelength of an order over σ^* reduces to the number of σ^* adjacencies it breaks. They therefore establish that the instrument has no intrinsic order preference and that it identifies the order that preserves the adjacencies of a chain, but not that it is equally sensitive to the dependence found in real corpora, where the context of a token comprises many tokens at many distances, some of which follow it in the original sequence.

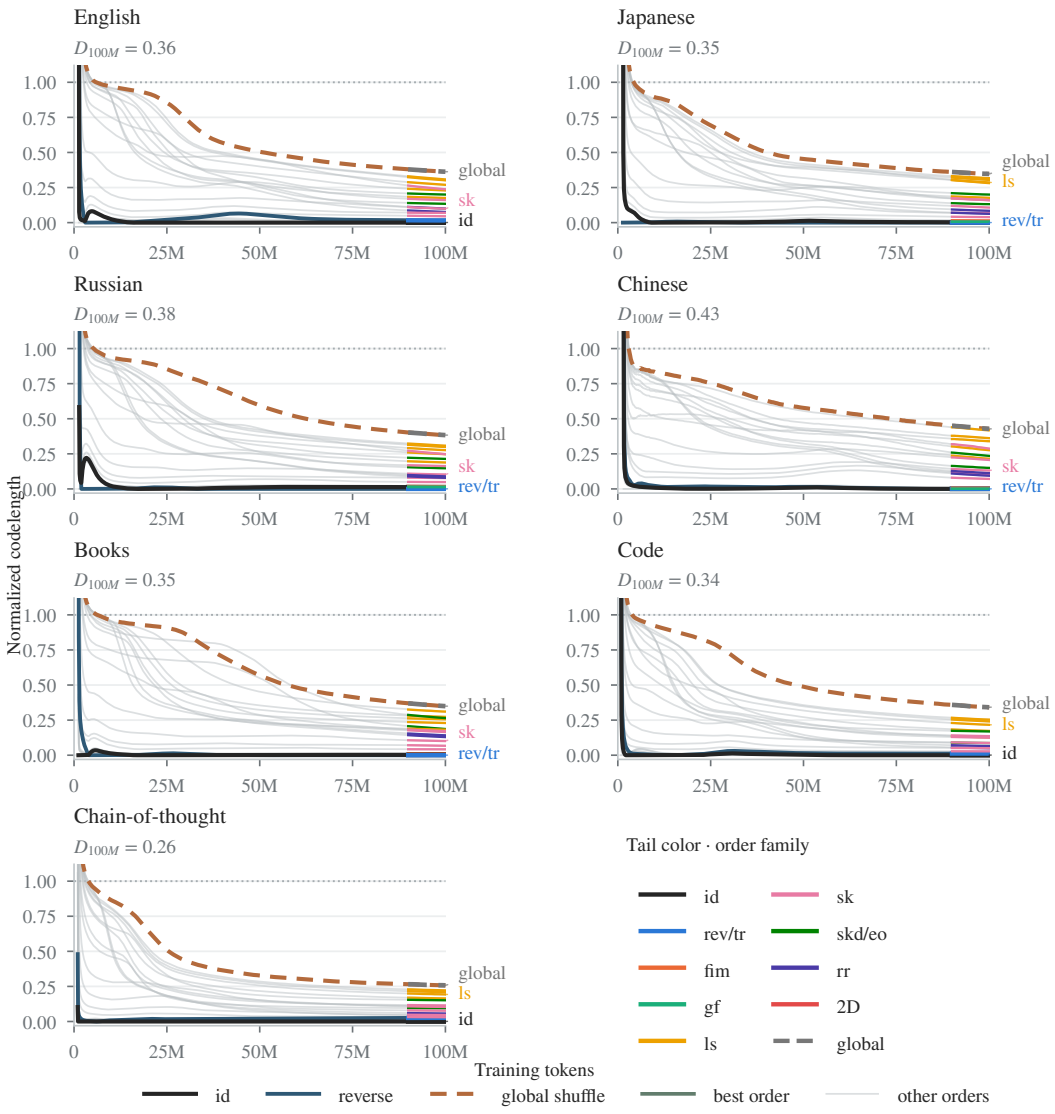

Figure 9: Normalized prequential codelength $(\tilde{\ell}_\sigma(N) - a(N))/(H_{\text{uni}} - a(N))$, where $\tilde{\ell}_\sigma$ is the 32-batch-half-life moving average and $a(N) = \min_\sigma \tilde{\ell}_\sigma(N)$. Zero is the best order at that budget, one is the unigram level, and the highest curve gives the relative gap D. This page shows text, code and reasoning; the next shows the remaining corpora. Curves use all recorded batches; nonpositive denominators are masked and initial values above 1.12 are clipped. The curve bodies highlight identity (black), reverse (blue), global shuffle (brown dashed), and the best order at 100M (green, unless already highlighted); other orders are gray. The final tenth of each observed curve uses order-family colors. Reversal includes tail reversal, and dyadic refinement shares a color with evens-then-odds. Selected labels identify an actual order's family, not a family average. Every curve ends at its measured 100M value, not a convergence estimate.

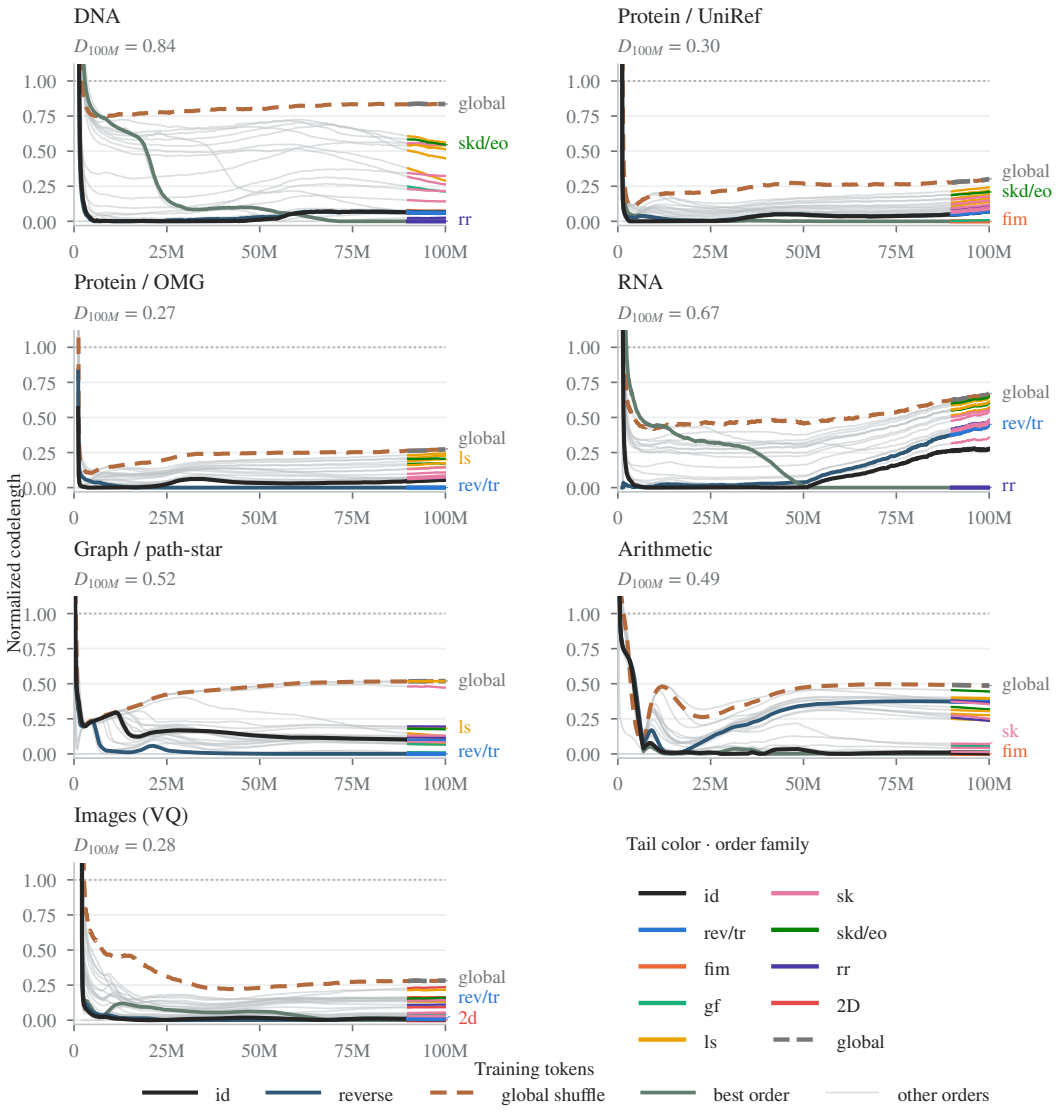

Figure 9 (continued): Normalized prequential codelength on biological sequences, path-star graphs, arithmetic and images. Both protein sources are shown separately. The normalization, scale and visual conventions are the same as on the preceding page.

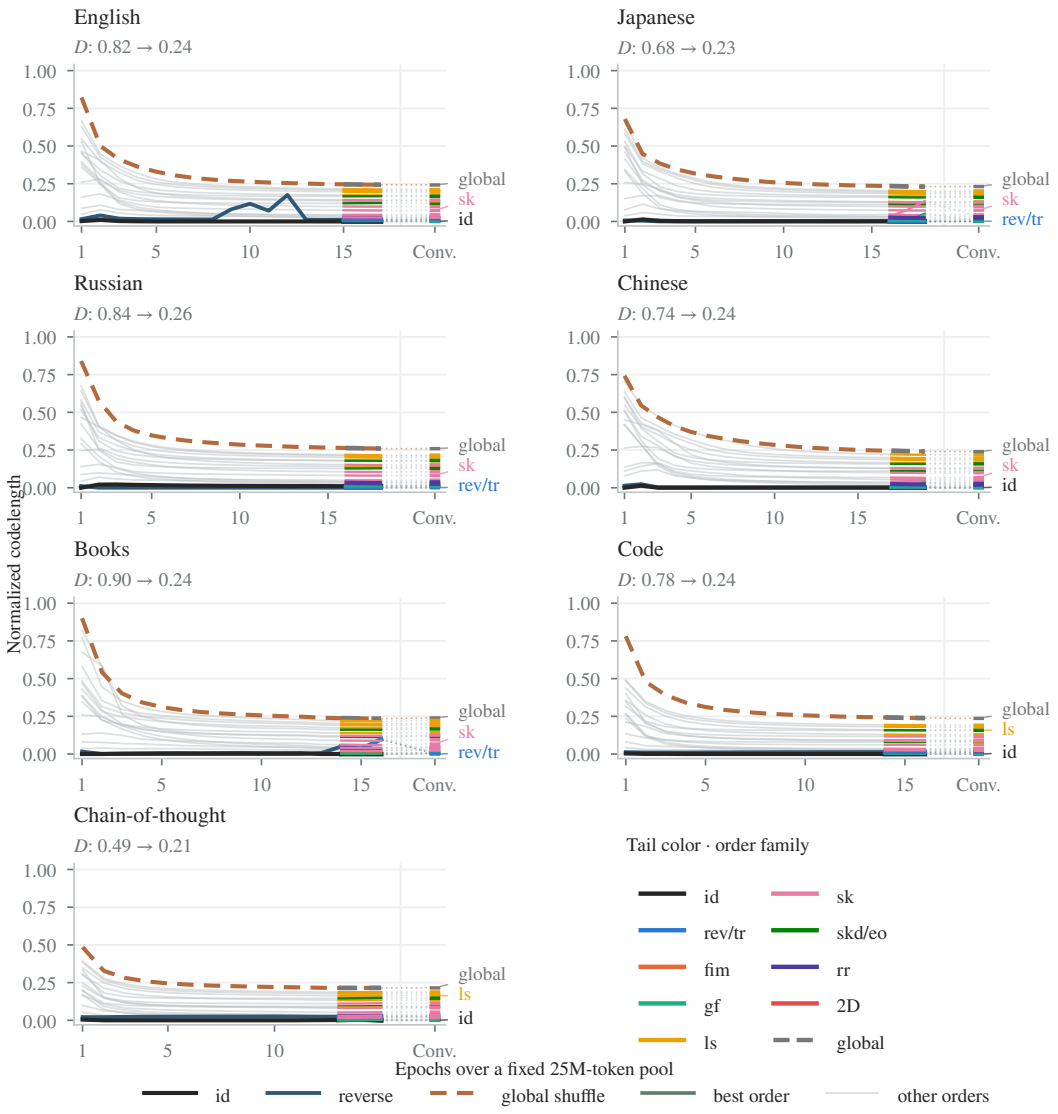

Figure 10: Convergence experiment on the fixed 25M-token pool. Held-out codelength at epoch e is normalized as $(\ell_\sigma^{\mathrm{ho}}(e) - a(e))/(H_{\mathrm{uni}} - a(e))$, with $a(e) = \min_\sigma \ell_\sigma^{\mathrm{ho}}(e)$; the top curve is the relative gap D. Panel subtitles give D after epoch 1 and at convergence. Curves average the available seeds (five initially) and stop at the last epoch reached by every order in at least three seeds. Dotted connectors lead to the separate *Conv.* column: each short horizontal tick is an order's converged value at its selected epoch, normalized using the lowest converged codelength. All ticks share one horizontal position and retain their exact vertical values. Curve-body styles and family-colored tails follow Figure 9, with green identifying the best converged order; convergence ticks also use family colors. This page shows text, code and reasoning, and the next the remaining corpora, on the same normalized scale.

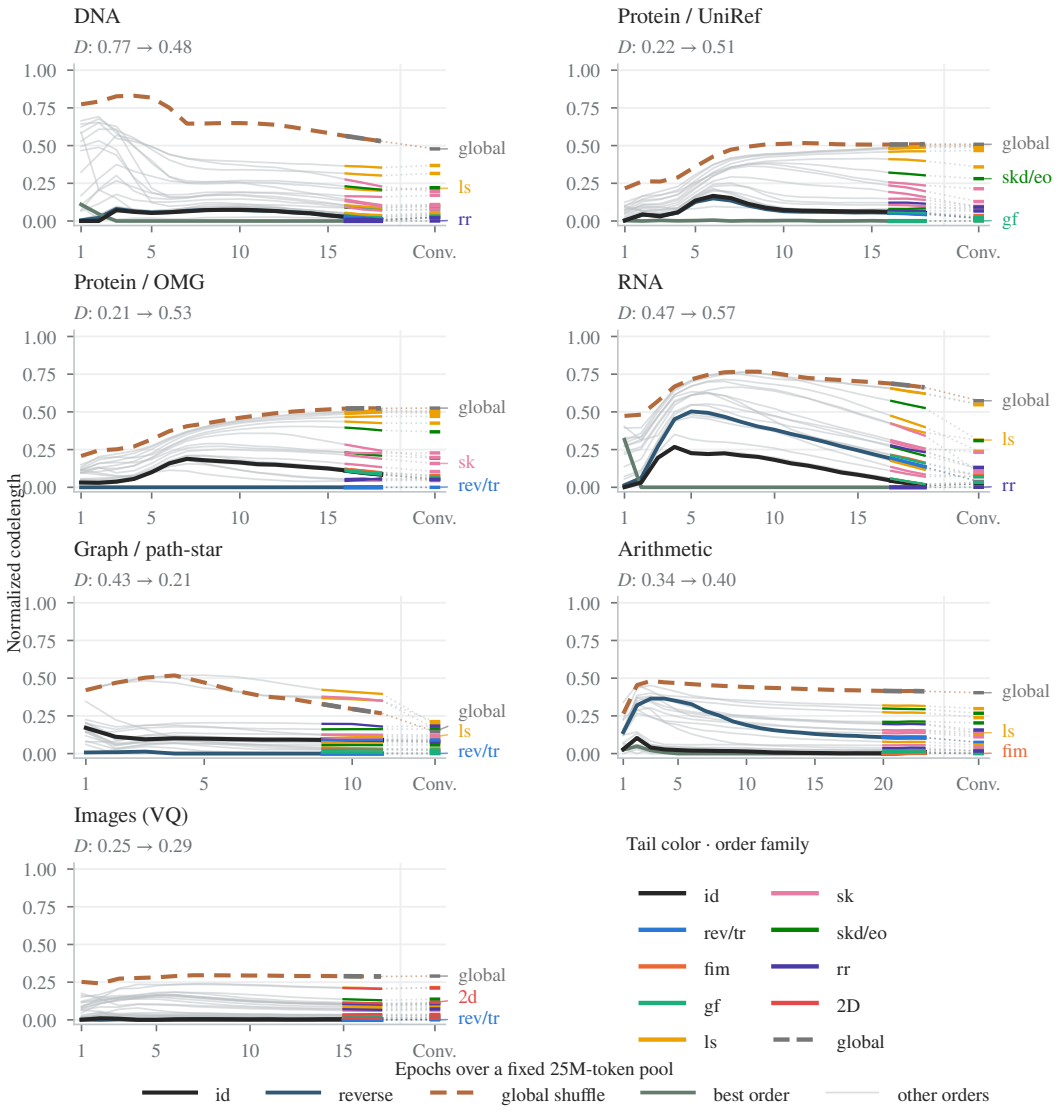

Figure 10 (continued): The convergence experiment on biological sequences, path-star graphs, arithmetic and images. Both protein sources are shown separately. All panels use the same normalization, scale and visual conventions as on the preceding page.

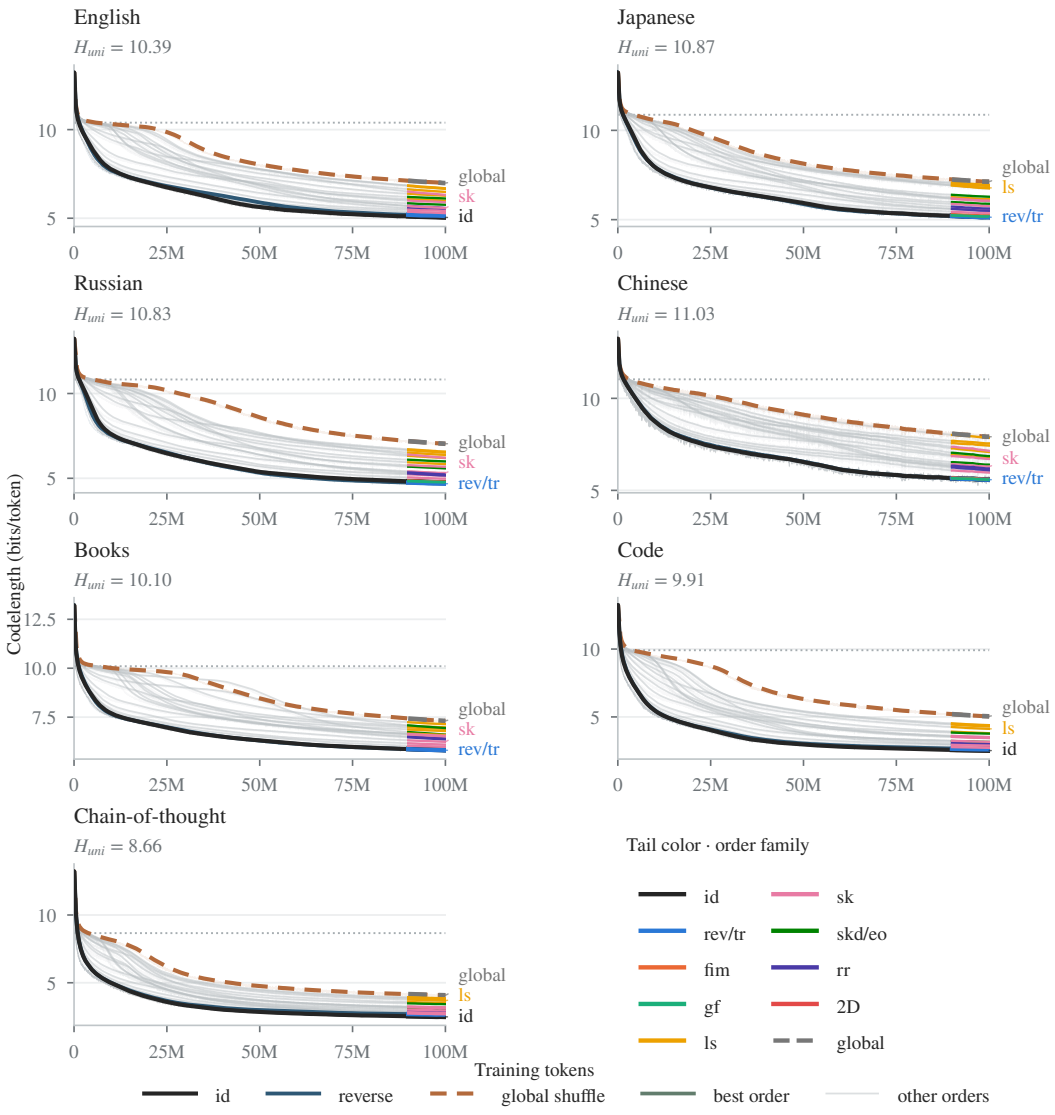

Figure 11: Prequential codelength of every order against training tokens N: the per-batch pre-update codelength ℓ_σ in faint lines and its moving average $\tilde{\ell}_\sigma$ (half-life 32 batches) in solid lines, both averaged over eight replicates. Every recorded batch is retained. The dotted horizontal reference is the unigram entropy H_{uni}; the dashed curve is the global shuffle. Curve-body styles, family-colored tails and selected family labels follow Figure 9. Endpoints are measured values at 100M, not convergence estimates. This page shows text, code and reasoning; the next shows the remaining corpora, including both protein sources. Vertical scales differ across panels because bits per token are not comparable across vocabularies.

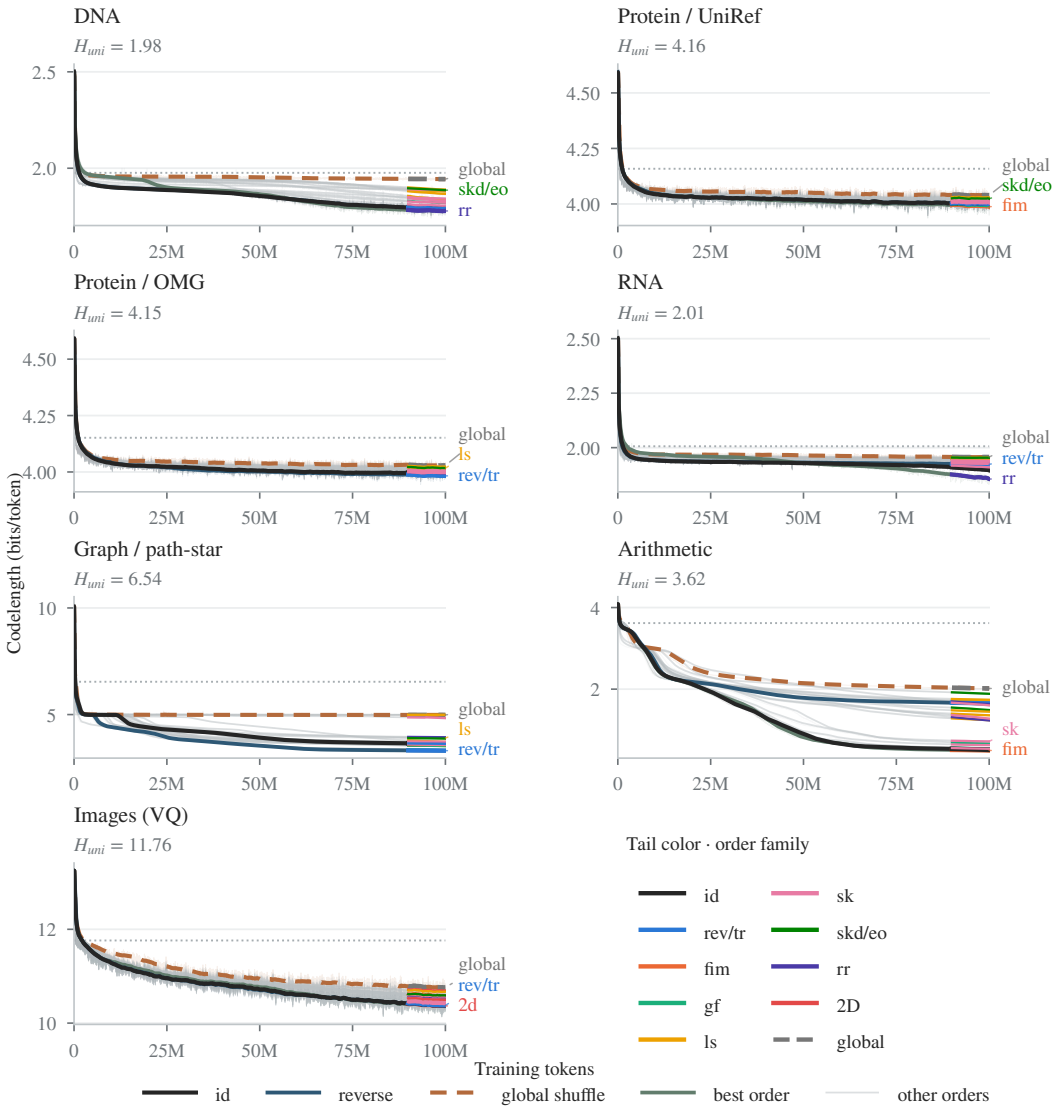

Figure 11 (continued): Prequential codelength on biological sequences, path-star graphs, arithmetic and images, including both protein sources. The per-batch curves, moving averages and visual conventions are the same as on the preceding page; vertical scales remain specific to each corpus.

Most data types are represented by a single corpus. Apart from natural language, represented by four Wikipedia editions, and protein, replicated on a second source, every data type enters the study as a single corpus. A contrast such as DNA versus protein therefore compares two corpora rather than two modalities, and apart from protein, where the change with training replicates across the two sources but the direction does not (Appendix E), no result has been shown to generalize across sources within a data type. The intervals are likewise conditional on the corpus instance and exclude corpus-sampling variance: between two independent draws of an anchor, codelengths differ by about 0.002 bits/token, which is below the narrowest margins we resolve on the corpora (0.0041 on UniRef and 0.0047 on OMG, Table 3, Appendix D), but not by a wide factor.