COMPOSELONG: SELF-SUPERVISED COMPOSITION OF LONG-CONTEXT TRAINING DATA FROM SHORT TEXTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective long-context training for large language models (LLMs) requires long training sequences with meaningful dependencies across distant parts of the context. Constructing such data presents two fundamental challenges: naturally occurring long documents are scarce, and document length alone does not guarantee useful long-range dependencies. Existing approaches address these challenges either by synthesizing long sequences from short texts or through quality-based filtering of existing long documents. However, both directions rely on hand-designed rules or criteria: synthesis methods do not learn from the structure of real long documents, while filtering methods retain only a fraction of the already limited supply of naturally occurring long documents. We introduce ComposeLong, a self-supervised framework that learns how to construct long-context training data using real long documents with strong long-range dependencies as supervision. Once trained, ComposeLong applies the learned patterns of document organization to large collections of short documents, synthesizing long-context training data without generating new text. ComposeLong outperforms prior synthesis methods on most reported metrics across RULER and LongBench v2.

1 INTRODUCTION

Equipping large language models (LLMs) with long-context capabilities is essential for applications such as long-document question answering and summarization, repository-level code understanding, and many-shot in-context learning (Gao et al., 2025d). Developing these capabilities typically requires continued pretraining on long sequences. Constructing effective long-context training data, however, presents two challenges. First, naturally occurring long documents suitable for long-context training are scarce and cover a limited range of domains. Second, document length alone does not guarantee meaningful dependencies across distant parts of the context. Randomly packing short documents creates long sequences but few useful relationships across document boundaries (Shi et al., 2024), while even naturally long documents may contain little information that requires long-distance context (Deng et al., 2026). The central challenge is therefore not simply to produce longer sequences, but to create meaningful long-range structure at scale.

Existing work addresses these challenges along two broad directions: 1) synthesis methods that compose new long sequences from shorter texts and 2) curation methods that select useful training examples from documents that are already long. Synthesis methods assemble short texts using semantic similarity or document structure (Shi et al., 2024; Gao et al., 2025a; Jia et al., 2026c; Zhang et al., 2025b), place distractors between related segments (Gao et al., 2025b), or use model-based verification to retain useful retrieved contexts (Jia et al., 2026a;b). These methods expand the supply and domain coverage of long training sequences, but rely on predefined rules to determine which retrieved texts are retained and how they are arranged. Curation methods instead score and filter existing long documents (Liu et al., 2024; Chen et al., 2024a; 2025; Wu et al., 2025; Deng et al., 2026). They can identify examples with useful distant dependencies, but cannot expand the supply of naturally occurring long documents and retain only a fraction of an already limited data pool. Even when language models participate in retrieval or verification, the construction of long-context training data remains governed by predefined criteria rather than learned from the structure of real long documents. These two directions leave an important opportunity unexplored: real long documents can serve not only as training examples, but also as supervision for a reusable composer. This

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.

motivates our central question: *can the organization of real long documents supervise a neural composer that constructs effective long-context training data from short-text corpora?*

We introduce ComposeLong, a self-supervised framework that uses real long documents with strong long-range dependencies as supervision for long-context data composition. Chunks within the same document provide examples of content that belongs together, while their original order shows how that content is organized. We use these signals to train an LLM-based composer without manual composition labels. Once trained, the composer transfers its learned policy to short-document corpora, selecting and arranging retrieved chunks into long-context training sequences while preserving the original text without generating new content. Retrieval supplies a tractable candidate set of document chunks, while the learned composer makes the final selection and ordering decisions. ComposeLong thus turns a limited collection of high-quality long documents into reusable supervision that can be applied to abundant short-text corpora. The supervision corpus can also provide a natural mechanism for adapting the composer to different domains or data preferences.

The resulting framework makes two main contributions:

1. To our knowledge, ComposeLong is the first self-supervised framework to learn chunk selection and ordering for long-context data synthesis from the organization of real long documents. The resulting composer transfers these learned patterns to short-text corpora, synthesizing long-context training data without generating new content.
2. ComposeLong outperforms prior synthesis methods on most reported metrics across RULER and LongBench v2. Controlled ablations further validate the learned composition framework and its main training and synthesis choices.

2 Related Work

The evolution of long-context LLMs. Early context-extension methods adapted pretrained short-context models through positional scaling or efficient fine-tuning. Position Interpolation (Chen et al., 2023) linearly rescales the position indices used by RoPE, extending LLaMA models from 2K to up to 32K, while YaRN (Peng et al., 2024) combines frequency-aware interpolation and attention scaling to reach 128K. PoSE (Zhu et al., 2024) exposes models to distant positions using short training sequences. LongLoRA (Chen et al., 2024b) reduces adaptation cost through shifted sparse attention and LoRA, and LongRoPE (Ding et al., 2024) searches for non-uniform RoPE scaling to extend models beyond two million tokens.

As context windows grew, research shifted toward continued pretraining on genuinely long sequences. Llama 2 Long (Xiong et al., 2024) upsamples long texts during continued pretraining to reach 32K. Fu et al. (2024) train on 1–5B tokens at 64K–80K and show that length upsampling and domain balance are both important for 128K performance. ProLong (Gao et al., 2025d) continues pretraining its base model for 20B tokens at 64K and another 20B at 512K, mixing books, code repositories, and short-context data. At million-token scale, Qwen2.5-1M (Yang et al., 2025b) progressively trains on natural and synthetic sequences up to 262K, then combines Dual Chunk Attention and YaRN to support 1M-token inference; Gemini 1.5 (Gemini Team, 2024) demonstrates strong retrieval at 1M tokens and reports experiments up to 10M. This progression makes the availability and quality of long training sequences a central bottleneck.

Defining and filtering high-quality long-context data. Long-context data quality has been measured at both the document and dependency levels. LongWanjuan (Liu et al., 2024) evaluates coherence, cohesion, and linguistic complexity, classifies texts as holistic, aggregated, or chaotic, and rebalances the retained categories. The ProLong data selector (Chen et al., 2024a) combines perplexity changes with dependency distance, weighting stronger dependencies at longer distances more heavily, and corrects for repetitive patterns. LongPPL (Fang et al., 2025) is a diagnostic metric rather than a corpus filter: it shows that average perplexity obscures the small set of tokens that depend on distant context and identifies them by contrasting long- and short-context predictions.

Attention-based filters more directly score distant interactions. LADM (Chen et al., 2025) aggregates span-level attention into a contextual dependency score and selects sequences with stronger cross-span dependencies. LongAttn (Wu et al., 2025) measures attention between sufficiently distant tokens and combines dependency strength with a distribution-uniformity correction. LongFilter (Deng et al.,

2026) instead defines quality as information gain by contrasting next-token predictions under long and short contexts, aggregating the token-level gains, and ranking existing long sequences.

Synthesizing long-context training data from short texts. Random packing concatenates short documents without creating explicit cross-document dependencies. Retrieval-based methods produce more connected sequences. SPLiCe (Staniszewski et al., 2025) collates related documents with BM25 or Contriever, while In-Context Pretraining (Shi et al., 2024) traverses a nearest-neighbor document graph. Quest (Gao et al., 2025a) groups documents using generated queries and keywords to balance relevance and diversity. LongPack (Zhuang et al., 2025) follows hyperlinks, while LiteLong (Jia et al., 2026c) uses topic information and retrieval to organize short texts into long-context training data.

Other methods construct or verify longer-range dependencies. LongSkywork (Zhao et al., 2024) interleaves chunks during continued pretraining. Untie the Knots (Tian et al., 2025) shuffles chunks, adds knot identifiers, and trains the model to recover their document structure. NExtLong (Gao et al., 2025b) inserts semantically similar hard negatives between ordered source chunks, preserving their dependencies while increasing the distance between them. EntropyLong (Jia et al., 2026a) retains contexts that reduce uncertainty at high-entropy positions, while PolicyLong (Jia et al., 2026b) repeats entropy estimation, retrieval, and verification with the evolving model to derive supporting contexts and hard negatives.

For long-context post-training, Re^{3}Syn (Zhang et al., 2025b) uses perplexity under both concatenation orders to identify directed dependencies and reorder retrieved documents. LongAlign (Bai et al., 2024), WildLong (Li et al., 2025), and LongMagpie (Gao et al., 2025c) synthesize long-context instruction–response examples for supervised adaptation. Across these approaches, text selection or ordering follows predefined similarity measures, external structures, verification criteria, or assembly rules. In contrast, ComposeLong learns a reusable composition policy from filtered real long documents and transfers it to short-text corpora.

3 METHOD

Following prior long-context data synthesis methods (Gao et al., 2025b; Jia et al., 2026b), we treat a *chunk* as the basic unit of composition. ComposeLong learns from the structure of real long documents and transfers the learned composition policy to short-text corpora. Given a consecutive chunk sequence $(c_0, c_1, \ldots, c_m)$ sampled from a real long document, we use c_0 as an *anchor* and combine the remaining chunks with distractors from other documents. The shuffled candidate pool is fed into an LLM-based composer, which is trained to recover the true continuation $(c_1, \ldots, c_m)$ in its original order. The original long document therefore provides self-supervision for both chunk selection and ordering.

At inference, we retrieve short documents with similar themes and split the source and retrieved documents into chunks. The first source chunk serves as the anchor, while all remaining chunks form the candidate pool. The retriever only proposes a tractable set of candidates; the composer decides which chunks to retain and how to order them. Fig. 1 provides an overview of composer training and long-document synthesis.

3.1 SELF-SUPERVISION FROM REAL LONG DOCUMENTS

Selecting supervision documents. Real long documents are the composer's only source of supervision, so their quality directly determines what the composer can learn. We therefore use LongFilter (Deng et al., 2026) to filter real long documents, retaining those with stronger long-range dependencies. For token i, let L_i^{long} and L_i^{short} denote its negative log-likelihood under long and local contexts, respectively. LongFilter assigns the token-level score

$$s_i = \exp\left(-L_i^{\text{long}}\right)\left(L_i^{\text{short}} - L_i^{\text{long}}\right). \tag{1}$$

The loss difference rewards tokens that are better predicted with long context, while $\exp(-L_i^{\text{long}})$ gives greater weight to tokens that the long-context model predicts confidently. We aggregate the token-level scores into a document score and retain high-scoring documents for composer training. Appendix B gives the complete scoring and selection procedure.

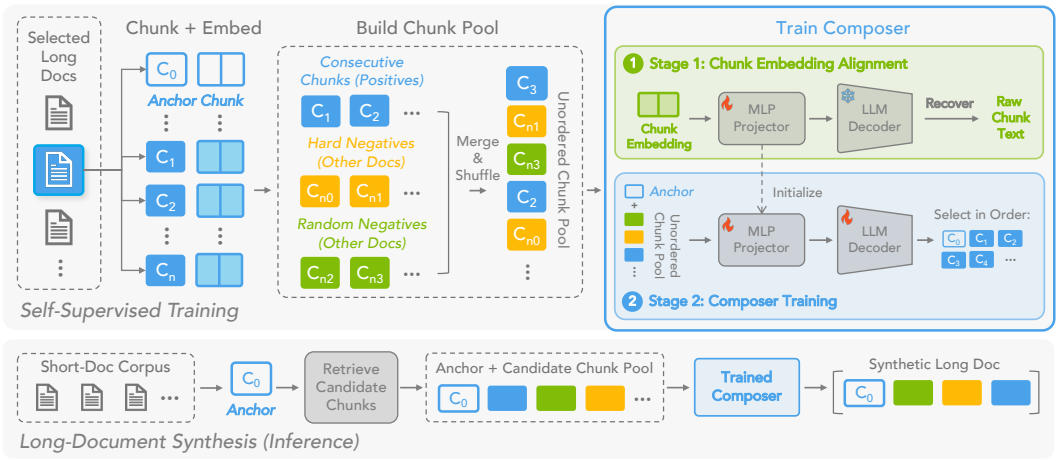


Figure 1: Overview of ComposeLong. For self-supervised training, LongFilter selects real long documents with strong long-range dependencies. We split each document into chunks and construct an unordered candidate pool by combining the ground-truth continuation with retrieved hard negatives and random negatives from other documents; the first chunk serves as the anchor. The LLM-based composer is trained in two stages: Stage 1 aligns chunk embeddings with the LLM through text reconstruction, and Stage 2 learns to autoregressively recover the ground-truth continuation from the shuffled pool. At inference, retrieval gathers candidate chunks from thematically related short documents, and the trained composer selects and orders them after the source anchor to synthesize a long document without generating new text.

Chunk representation and continuation construction. During composer training, we divide each selected document into an ordered sequence of non-overlapping chunks and represent each chunk c_i by an embedding $\mathbf{e}_i$. To form a training example, we sample a contiguous span $\mathcal{S} = (c_0, c_1, \ldots, c_m)$, use c_0 as the anchor, and treat $(c_1, \ldots, c_m)$ as the ground-truth continuation.

Candidate pool construction. Given the sampled span $\mathcal{S}$, we construct a candidate pool $\mathcal{C}$ by combining the ground-truth continuation $(c_1, \ldots, c_m)$ with distractor chunks, while keeping the anchor c_0 separate as conditioning context. The distractors consist of hard negatives and random negatives. We obtain hard negatives through neighbor retrieval, using the anchor or other chunks from its source document as queries, and sample random negatives directly from the corpus to increase the diversity of the candidate pool. All distractors come from documents other than the source document of $\mathcal{S}$. We remove duplicate chunks and shuffle the resulting candidate pool so that its input order does not reveal the answer. The sampled span $\mathcal{S}$ then provides the supervision: the composer learns to select $(c_1, \ldots, c_m)$ from $\mathcal{C}$ and recover their original order.

3.2 LLM-Based Composer

The composer operates on precomputed chunk embeddings and consists of a two-layer MLP projector and a decoder-only LLM. We follow the architecture and two-stage training procedure of PE-Rank (Liu et al., 2025), adopting embedding reconstruction for projector alignment and autoregressive ranking over candidate embeddings.

Stage 1: embedding alignment. The chunk embeddings $\mathbf{e}_i$ and the LLM's token embeddings lie in different representation spaces. To bridge them, we use a two-layer MLP projector to map each $\mathbf{e}_i$ to the corresponding LLM-space representation $\mathbf{z}_i$. We then feed $\mathbf{z}_i$ into the frozen LLM and train the projector with a causal language modeling objective to reconstruct the raw text of chunk c_i. Because this reconstruction objective only aligns the two representation spaces, it can use any available chunks without requiring long-document supervision.

Stage 2: anchor-conditioned composition. Stage 2 teaches the composer to select the ground-truth continuation from the candidate pool and recover its order. We initialize the projector from Stage 1 and train it jointly with the LLM decoder. Given the anchor c_0 and the shuffled candidate pool $\mathcal{C}$, the

composer receives the anchor representation $\mathbf{z}_0$ and the candidate representations $\{\mathbf{z}_c\}_{c\in\mathcal{C}}$. Starting from the anchor, it scores the candidate representations and selects their corresponding chunks one at a time, with $(c_1, \ldots, c_m)$ as the target sequence.

We first define the ranking objective under clean teacher forcing. Before predicting c_t, the composer is given the correct preceding chunks $c_{<t} = (c_1, \ldots, c_{t-1})$. The LLM processes the anchor, the candidate pool, and this ordered prefix to produce a hidden state $\mathbf{h}_t$. Let $\mathcal{U}_t = \mathcal{C} \setminus \{c_1, \ldots, c_{t-1}\}$ denote the candidates that remain available. Following PE-Rank (Liu et al., 2025), we score these candidates directly in the embedding space. The probability that a remaining candidate c is the next chunk is

$$p_t(c) = \frac{\exp\big(\mathbf{h}_t^\top \mathbf{z}_c\big)}{\sum_{c'\in\mathcal{U}_t} \exp\big(\mathbf{h}_t^\top \mathbf{z}_{c'}\big)}, \qquad c \in \mathcal{U}_t. \tag{2}$$

The dot product measures how well each remaining chunk fits the next position, and the softmax turns these scores into a distribution over $\mathcal{U}_t$. The training target at step t is the next ground-truth chunk c_t. We optimize the complete continuation with an embedding-based ListMLE-style objective (Xia et al., 2008),

$$\mathcal{L}_{\text{rank}} = -\frac{1}{m}\sum_{t=1}^{m} \log p_t(c_t). \tag{3}$$

Comparing c_t with distractors teaches the composer which chunks belong to the continuation, while comparing it with later ground-truth chunks teaches which continuation chunk comes next. We do not use PE-Rank's auxiliary passage-text loss or KL-distillation loss.

Robustness to imperfect histories. In the clean objective above, every ground-truth continuation chunk is included in $\mathcal{C}$, and the composer observes only correct prefixes. At inference, however, the composer operates on chunks retrieved from a short-text corpus: there is no ground-truth continuation, and the candidate pool may not contain an ideal next chunk. The composer may therefore need to continue after an imperfect selection, whose effect can accumulate over later steps.

To reduce this train–inference gap, we train on a mixture of clean and corrupted prefixes. For some examples, we randomly replace chunks in the teacher-forced prefix $c_{<t}$ with either sampled distractors from $\mathcal{C}$ or the composer's own incorrect predictions. Sampled distractors provide diverse deviations from the ground-truth sequence, while model-generated errors more closely resemble the selections encountered during inference. An initial forward pass without gradient computation supplies these model-generated errors; a second forward pass processes the resulting prefix while keeping the anchor and ground-truth targets unchanged. This teaches the composer to resume the correct continuation after an imperfect earlier selection without requiring a full autoregressive rollout during training. Appendix A reports the corruption probabilities.

3.3 Long-Document Synthesis from Short Texts

After training, we apply the composer to a corpus of short documents. Given a short source document, we retrieve documents with similar themes and split both the source and retrieved documents into chunks. We process the source chunks in their original order, using each one as the anchor for an independent composition trajectory. For each anchor, the retrieved chunks form a shuffled candidate pool, from which the composer autoregressively selects an ordered continuation. We then concatenate the anchor–continuation groups in source order to produce a long document without generating new content. Appendix C gives the retrieval and candidate-construction details.

4 Experiments

We evaluate whether long documents composed by ComposeLong provide more effective long-context pretraining data than documents constructed with predefined rules. We examine performance across context lengths and whether the benefits persist after instruction tuning. The source corpora, training-token budget, and optimization settings are held fixed within each comparison.

4.1 Experimental Setup

Datasets. We use separate corpora to train the composer and to synthesize downstream pretraining data. Following NExtLong (Gao et al., 2025b), we chunk each document with a paragraph-grouping procedure that targets 2,048 characters per chunk while keeping oversized paragraphs intact. Tab. 1 reports the resulting corpus statistics.

For composer training, we take documents with at least 250K characters from SlimPajama-627B (Soboleva et al., 2023), documents with at least 128K tokens from Dolma3 LongMiNo (Team OLMo et al., 2025), and 512K-token The Stack sequences released by ProLong (Gao et al., 2025d). We apply LongFilter (Deng et al., 2026) separately to SlimPajama and Dolma3 and retain the top 30% of each corpus by score. We retain all 6.0K The Stack sequences from ProLong without further filtering because this collection is comparatively small. Only retained documents provide ground-truth continuations. When constructing a candidate pool, chunks from both retained and non-retained documents may serve as distractors, except those from the current source document. Appendix B describes the scoring and selection details.

Table 1: Composer-training and synthesis corpora.

Dataset	Docs	Chunks
Training		
SlimPajama-627B	88.0K	26.6M
Dolma3 LongMiNo	168.7K	168.2M
ProLong The Stack	6.0K	5.3M
Synthesis		
FineWeb-Edu	190.2M	557.0M
Cosmopedia-v2	39.1M	94.7M

Following prior long-context data synthesis work (Gao et al., 2025b; Jia et al., 2026a;c), we construct the synthesis pool by uniformly sampling a total of 32,000 short documents from the combined FineWeb-Edu (Penedo et al., 2024) and Cosmopedia-v2 (Ben Allal et al., 2024) pool.

Composer training. We encode chunks with Qwen3-Embedding-0.6B (Zhang et al., 2025a) and use Qwen3-4B (Yang et al., 2025a) as the LLM backbone of the composer. In Stage 1, we freeze the LLM and train the projector to reconstruct up to the first 2,048 tokens of each chunk from its embedding, followed by an end-of-sequence token. In Stage 2, each training example contains 128–448 consecutive ground-truth continuation chunks, 128–1,280 retrieved hard negatives, and 64–192 random negatives, for at most 1,920 candidates. We enable history corruption for 50% of Stage 2 examples and corrupt each eligible history position in those examples with probability 30%. Each corrupted position uses either a sampled distractor or, with equal probability when available, an incorrect prediction from a no-gradient forward pass. The resulting expected corruption rate is approximately 15% across eligible positions. Appendix A gives the complete training settings.

Document synthesis. Following NExtLong (Gao et al., 2025b), we use each chunk of the source document, in its original order, as the anchor of an independent composition trajectory. For each anchor, we form a pool of 1,280 candidate chunks from thematically related documents retrieved with document-level BM25, following LiteLong (Jia et al., 2026c). We exclude chunks selected for earlier anchors in the same output and shuffle each candidate pool before composition. The composer selects and orders a continuation from this pool, and we assemble the resulting anchor–continuation groups in source order to construct a document of approximately 128K tokens. We compare this per-chunk strategy with document-level synthesis using a shared candidate pool in Tab. 4 and Sec. 4.3. Appendix C describes retrieval and candidate construction.

Long-context LLM training. We evaluate the synthesized data by continuing pretraining of Qwen2.5-3B (Qwen et al., 2024). We use YaRN (Peng et al., 2024) with a scaling factor of 4 to extend its native 32K context window to 128K, and evaluate RULER directly after continued pretraining. For LongBench v2, we further fine-tune each checkpoint following the LongMagpie training setup (Gao et al., 2025c). Appendix A reports the complete continued-pretraining and fine-tuning configurations.

4.2 Main Results

We evaluate the synthesized data in two complementary settings. RULER measures how performance changes with context length under a controlled Qwen2.5-3B continued-pretraining setup. LongBench v2 evaluates whether the effect of the pretraining data persists after instruction tuning.

RULER on Qwen2.5-3B. We evaluate ComposeLong and three prior long-context data synthesis methods on RULER (Hsieh et al., 2024): NExtLong (Gao et al., 2025b), EntropyLong (Jia et al.,

2026a), and LiteLong (Jia et al., 2026c). We use the officially released dataset for each baseline and continue pretraining Qwen2.5-3B (Qwen et al., 2024) on each dataset.

We do not include PolicyLong (Jia et al., 2026b) for two reasons. First, it uses iterative on-policy synthesis, whereas our comparison focuses on off-policy methods that construct a fixed dataset before model training. Second, its training data were not publicly available at the time of our study. We also do not reuse the baseline scores reported by PolicyLong because its paper does not provide the base-model training and evaluation details needed to reproduce those results. For a fair comparison, we retrain all available baselines and evaluate the resulting models ourselves with EleutherAI's `lm-evaluation-harness` (Gao et al., 2023) under the official RULER protocol; Appendix D gives the details.

Table 2: RULER performance of Qwen2.5-3B models continually pretrained on long-context data produced by different construction methods. Scores are reported at each context length and averaged across lengths.

Training data	8K	16K	32K	64K	128K	Avg.
NExtLong	77.57	70.42	66.68	60.92	58.05	66.73
EntropyLong	79.59	74.05	69.88	67.48	61.60	70.52
LiteLong	81.09	76.40	72.04	67.71	62.33	71.91
ComposeLong (Ours)	**82.17**	**77.25**	**72.98**	**69.06**	**65.07**	**73.31**

Tab. 2 shows that ComposeLong outperforms all three baselines at every context length. The gain is largest at 128K, where ComposeLong reaches 65.07 compared with 62.33 for LiteLong, suggesting that its benefit is strongest at the longest evaluated context.

LongBench v2 after instruction tuning. LongBench v2 (Bai et al., 2025) evaluates whether the advantage of ComposeLong's continued-pretraining data remains after supervised adaptation. We apply the same LongMagpie (Gao et al., 2025c) instruction-tuning configuration to every Qwen2.5-3B checkpoint obtained after continued pretraining and evaluate all models on the complete set of 503 examples without chain-of-thought prompting or retrieval augmentation. Appendix D gives the complete evaluation protocol.

Table 3: LongBench v2 accuracy (%) of Qwen2.5-3B models after continued pretraining on each method's data, followed by the same LongMagpie (Gao et al., 2025c) instruction tuning. Results are reported overall and by input length and difficulty. Bold and underlined values indicate the best and second-best results in each column, respectively.

Model	Short	Medium	Long	Easy	Hard	Overall
NExtLong	32.22	**25.12**	<u>27.78</u>	31.77	**26.05**	<u>28.23</u>
EntropyLong	<u>33.33</u>	<u>24.19</u>	25.00	<u>32.81</u>	24.44	27.63
LiteLong	32.22	21.40	**30.56**	32.29	24.12	27.24
ComposeLong (Ours)	**35.56**	**25.12**	<u>27.78</u>	**35.42**	<u>25.72</u>	**29.42**

Tab. 3 shows that ComposeLong performs competitively after instruction tuning. It achieves the highest overall accuracy and the best results on the short and easy subsets, while matching NExtLong on medium-length examples. On the long and hard subsets, its performance remains comparable to the strongest method, although LiteLong and NExtLong perform best, respectively. Overall, these results indicate that the benefit of ComposeLong's continued-pretraining data remains observable after a shared instruction-tuning stage.

4.3 Ablation Studies and Analysis

We analyze the composer's synthesis capacity and isolate the effects of history corruption, learned continuation order, LongFilter-selected supervision, inference-pool retrieval, and per-chunk synthesis. Tab. 4 summarizes the component ablations, while Fig. 2 and Tab. 5 analyze inference-pool retrieval.

Synthesis length capacity. Each chunk occupies one embedding position in the composer. For one autoregressive trajectory, let L be the composer backbone context length, P the positions used by the prompt and anchor, and K the number of candidates. The pool occupies K positions and each selection adds one history position, giving $M_{\max} \leq \min(K, L - P - K)$ selected chunks. With average chunk length $\bar{\ell}_{\text{char}}$ in characters and character-to-token ratio ρ, the continuation length is approximately $T_{\max} \approx M_{\max}\bar{\ell}_{\text{char}}/\rho$.

Following NExtLong (Gao et al., 2025b), our newline-aware chunker targets approximately 2,048 characters while preserving paragraph boundaries. With the Qwen3-4B backbone's 32K context (Yang et al., 2025a) and $K = 1{,}280$ inference candidates, the candidate count is the binding constraint. Our synthesized documents average 4.79 characters per Qwen3 token, giving a nominal single-trajectory capacity of approximately 0.55M text tokens. The actual synthesized length is controlled by the per-anchor budget described in Appendix C and varies with the realized chunk lengths. The full pipeline combines independent trajectories for ordered source chunks, so final document length also depends on the number of anchors and the assembly target. A longer-context backbone increases trajectory capacity only with a correspondingly larger candidate pool.

Table 4: Ablations of ComposeLong on RULER. Rows prefixed with − remove the corresponding component or design choice from the full method. Bold and underlined values indicate the best and second-best results in each column, respectively.

Composer variant	8K	16K	32K	64K	128K	Avg.
ComposeLong (full)	<u>82.17</u>	<u>77.25</u>	**72.98**	**69.06**	**65.07**	**73.31**
− *History corruption*	79.80	74.63	71.83	66.82	62.68	71.15
− *Learned order*	80.43	73.73	71.25	66.27	61.78	70.69
− *LongFilter selection*	**82.31**	**77.37**	<u>72.82</u>	<u>69.01</u>	<u>64.87</u>	<u>73.28</u>
− *Per-chunk synthesis*	81.02	74.52	70.63	64.99	61.36	70.50

Robustness to imperfect histories. During autoregressive composition, an incorrect selection becomes part of the history used for later decisions. Training only on ground-truth histories therefore creates a mismatch with inference, where the composer must continue from its own previous selections. To evaluate the robustness strategy in Sec. 3.2, we compare the full composer with a variant trained using teacher forcing alone, without sampled-distractor or model-generated history corruption. As shown in Tab. 4, history corruption improves RULER at every evaluated context length. These gains suggest that exposure to imperfect histories improves robustness to earlier selection errors during autoregressive composition.

Effect of learned continuation order. The composer determines both which chunks to select and how to order them after each anchor. To isolate the effect of the learned order, we keep each anchor and the continuation chunks selected by the full composer fixed, but randomly shuffle the chunks within each continuation. This preserves the selected content and training-data volume while removing the composer-predicted order. Tab. 4 shows that shuffling reduces RULER performance at every evaluated context length. The consistent decrease suggests that the learned continuation order provides benefits beyond chunk selection alone.

Effect of LongFilter-selected supervision. LongFilter determines which real long documents provide ground-truth continuations for composer training. To isolate its effect, we replace the selected documents with an equally sized random sample from the unfiltered long-document pool, while keeping composer training, synthesis, and downstream continued pretraining fixed. As shown in Tab. 4, the two variants perform similarly overall, but LongFilter-selected supervision is consistently better at 32K, 64K, and 128K. We expect a stronger composer backbone to make better use of this supervision and further widen the gap at longer context lengths.

Effect of per-chunk synthesis. Following NExtLong (Gao et al., 2025b), our full pipeline uses each source chunk as the anchor of an independent composition trajectory and assembles the resulting anchor–continuation groups in source order. This preserves document-level ordering while decomposing long-document synthesis into local composition decisions. We compare it with a document-level variant that treats the complete source document as one synthesis instance and uses a shared candidate pool. Tab. 4 shows that per-chunk synthesis performs better at every evaluated

context length, with larger differences at 64K and 128K. These results suggest that source-level ordering and local anchor-conditioned composition provide complementary structure.

Document-level versus chunk-level retrieval.

Prior work has explored chunk-level embedding retrieval with FAISS in NExtLong (Gao et al., 2025b) and document-level BM25 retrieval in LiteLong (Jia et al., 2026c). We study how these two retrieval strategies affect the synthesis stage of ComposeLong. Because the composer learns from the organization of real long documents, we seek an inference candidate pool whose chunk relationships resemble those found within real long documents. Our full configuration uses BM25 to retrieve thematically related documents and places their chunks in the candidate pool. We compare it with chunk-level FAISS retrieval, which directly returns the nearest chunk embeddings, while keeping the pool size, composer, synthesis procedure, and downstream training fixed.

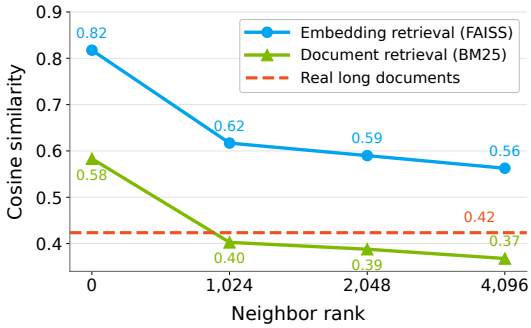


Figure 2: Anchor–candidate cosine similarity by neighbor rank. FAISS directly retrieves chunk neighbors, whereas BM25 similarity is averaged over chunks from retrieved documents. The dashed line shows the mean within-document chunk similarity of real long documents.

Fig. 2 shows that document-level BM25 produces anchor–candidate similarities close to the within-document similarity observed in real long documents. In contrast, the highest-ranked FAISS neighbors are substantially more similar to the anchor, producing a semantically concentrated pool with more repetitive content and less diversity within the same theme. As retrieval depth increases, FAISS similarity decreases, indicating that lower-ranked neighbors increasingly deviate from the anchor. Document-level retrieval provides a better balance by retrieving documents with related themes while preserving diversity among their constituent chunks. Consistent with this analysis, Tab. 5 shows that BM25 outperforms chunk-level FAISS at every evaluated context length.

Table 5: Effect of candidate-pool retrieval during ComposeLong synthesis on RULER.

Retrieval method	8K	16K	32K	64K	128K	Avg.
Document-level BM25	**82.17**	**77.25**	**72.98**	**69.06**	**65.07**	**73.31**
Chunk-level FAISS	79.80	74.20	70.97	66.75	62.51	70.84

5 Conclusion

We introduced ComposeLong, a self-supervised framework that trains a neural composer for long-context data synthesis. Rather than defining the semantic and structural properties of useful long-context training data through hand-designed rules, ComposeLong learns how to select and order chunks from real long documents with strong long-range dependencies. The trained composer transfers these patterns to short-text corpora, assembling existing chunks into long training documents without generating new text. On Qwen2.5-3B, ComposeLong outperforms prior synthesis methods on most reported metrics across RULER and LongBench v2. Under our evaluation settings, these results show that learning a composition strategy from real long documents is more effective than relying on fixed, hand-designed synthesis rules. They also demonstrate that real long documents can serve as reusable supervision for learning long-context data composition.

Limitations. ComposeLong requires a dedicated training process for the composer, introducing additional computational overhead before data synthesis. Its synthesis quality also depends on a sufficiently relevant and diverse candidate pool. Our current pipeline constructs this pool through neighbor retrieval, so the composer cannot select useful chunks that retrieval fails to provide, limiting the quality and diversity of the synthesized data. We leave reducing the training overhead and developing more effective candidate-pool construction methods to future work.

AI USE STATEMENT

We used generative AI tools for the following tasks:

- **Provide feedback on experimental design and ablation studies:** AI tools were used to discuss experimental settings and suggest or refine ablation studies. The authors made all final decisions about the experimental design.
- **Implement methods and process data:** Coding agents were used to implement the training and inference pipelines for ComposeLong, as well as the associated data-processing code. The authors reviewed and tested the implementations and verified their outputs.
- **Assist with result interpretation:** AI tools helped compare results across evaluation settings and draft possible interpretations. The authors verified all reported values against the original experimental outputs and reviewed the resulting conclusions.
- **Revise the structure and writing of the paper:** AI tools assisted with organizing sections and improving the clarity and readability of the text. All AI-assisted writing was reviewed and revised by the authors.
- **Identify, summarize, and verify related literature:** AI tools assisted with literature search, summarization, and reference checking. The authors checked literature claims and citations against the corresponding papers or official sources.
- **Create or revise tables, figures, and LaTeX artifacts:** AI tools assisted with preparing and formatting tables, figures, and LaTeX source. The authors verified the underlying content and the final rendered artifacts.

We have reviewed and validated all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, results, and artifacts.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. Longalign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.74. URL https://aclanthology.org/2024.findings-emnlp.74/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Loubna Ben Allal, Anton Lozhkov, and Daniel van Strien. Cosmopedia: How to create large-scale synthetic data for pre-training. Hugging Face Blog, March 2024. URL https://huggingface.co/blog/cosmopedia.

Jianghao Chen, Junhong Wu, Yangyifan Xu, and Jiajun Zhang. LADM: Long-context training data selection with attention-based dependency measurement for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3076–3090. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.154. URL https://aclanthology.org/2025.acl-long.154/.

Longze Chen, Ziqiang Liu, Wanwei He, Yinhe Zheng, Hao Sun, Yunshui Li, Run Luo, and Min Yang. Long context is not long at all: A prospector of long-dependency data for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8222–8234. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.acl-long.447. URL https://aclanthology.org/2024.acl-long.447/.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. URL https://arxiv.org/abs/2306.15595.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8220–8238, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/211ab571cc9f3802afa6ffff52ae3e5b-Abstract-Conference.html.

Haoran Deng, Yingyu Lin, Zhenghao Lin, Xiao Liu, Yizhou Sun, Yian Ma, and Yeyun Gong. Beyond length: Quantifying long-range information for long-context LLM pretraining data. In *International Conference on Learning Representations*, volume 2026, pp. 12983–12995, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/15cc66a412c9fcb0c41050904da4afa9-Abstract-Conference.html.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11091–11104. PMLR, 2024. URL https://proceedings.mlr.press/v235/ding24i.html.

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*, volume 2025, pp. 94541–94563, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ebd6641c32ed633c2a3addc689d39896-Abstract-Conference.html.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14125–14134. PMLR, 2024. URL https://proceedings.mlr.press/v235/fu24d.html.

Chaochen Gao, Xing Wu, Qi Fu, and Songlin Hu. Quest: Query-centric data synthesis approach for long-context scaling of large language model. In *International Conference on Learning Representations*, volume 2025, pp. 91812–91832, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e54e6eef11f87a874bf1e4551fc6d04e-Abstract-Conference.html.

Chaochen Gao, Xing Wu, Zijia Lin, Debing Zhang, and Songlin Hu. NExtlong: Toward effective long-context training without long documents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18566–18584. PMLR, 2025b. URL https://proceedings.mlr.press/v267/gao25n.html.

Chaochen Gao, Xing Wu, Zijia Lin, Debing Zhang, and Songlin Hu. Longmagpie: A self-synthesis method for generating large-scale long-context instructions. In *Advances in Neural Information Processing Systems*, volume 38, pp. 12866–12889, 2025c. doi: 10.52202/085713-0434. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1336245d8430ff3e3c8a7b67d1a2fcc4-Abstract-Conference.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. URL https://zenodo.org/records/10256836.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7376–7399. Association for Computational Linguistics, 2025d. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. URL https://arxiv.org/abs/2403.05530.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Junlong Jia, Ziyang Chen, Xing Wu, Chaochen Gao, Zijia Lin, Songlin Hu, and Binghui Guo. Entropylong: Effective long-context training via predictive uncertainty. In *International Conference on Learning Representations*, volume 2026, pp. 4046–4068, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/07e278a120830b10aae20cc600a8c07b-Abstract-Conference.html.

Junlong Jia, Ziyang Chen, Xing Wu, Chaochen Gao, TingHao Yu, Feng Zhang, Songlin Hu, and Binghui Guo. Policylong: Towards on-policy context extension. In *Third Conference on Language Modeling*, 2026b. URL https://colm.eventhosts.cc/Conferences/2026/AcceptedPapers. arXiv:2604.07809.

Junlong Jia, Xing Wu, Chaochen Gao, Ziyang Chen, Zijia Lin, Zhongzhi Li, Weinong Wang, Haotian Xu, Donghui Jin, Debing Zhang, and Binghui Guo. Litelong: Resource-efficient long-context data synthesis for LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31274–31282, 2026c. doi: 10.1609/aaai.v40i37.40390. URL https://ojs.aaai.org/index.php/AAAI/article/view/40390.

Jiaxi Li, Xingxing Zhang, Xun Wang, Xiaolong Huang, Li Dong, Liang Wang, Si-Qing Chen, Wei Lu, and Furu Wei. Wildlong: Synthesizing realistic long-context instruction data at scale. *arXiv preprint arXiv:2502.16684*, 2025. URL https://arxiv.org/abs/2502.16684.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. Leveraging passage embeddings for efficient listwise reranking with large language models. In *Proceedings of the ACM on Web Conference 2025*, pp. 4274–4283, 2025.

Xiaoran Liu, Kai Lv, Qipeng Guo, Hang Yan, Conghui He, Xipeng Qiu, and Dahua Lin. Longwanjuan: Towards systematic measurement for long text quality. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5709–5725. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.327. URL https://aclanthology.org/2024.findings-emnlp.327/.

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*, volume 37, 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 31932–31951, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/874a4d89f2d04b4bcf9a2c19545cf040-Abstract-Conference.html.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Gergely Szilvasy, Rich James, Xi Victoria Lin, Noah A. Smith, Luke Zettlemoyer, Scott Yih, and

Mike Lewis. In-context pretraining: Language modeling beyond document boundaries. In *International Conference on Learning Representations*, volume 2024, pp. 37348–37361, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/a1fe2365abdb691332537990a6385909-Abstract-Conference.html.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Konrad Staniszewski, Szymon Tworkowski, Sebastian Jaszczur, Yu Zhao, Henryk Michalewski, Łukasz Kuciński, and Piotr Miłoś. Structured packing in LLM training improves long context utilization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25201–25209, 2025. doi: 10.1609/aaai.v39i24.34706. URL https://ojs.aaai.org/index.php/AAAI/article/view/34706.

Team OLMo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. OLMo 3. *arXiv preprint arXiv:2512.13961*, 2025. URL https://arxiv.org/abs/2512.13961.

Junfeng Tian, Da Zheng, Yang Chen, Rui Wang, Colin Zhang, and Debing Zhang. Untie the knots: An efficient data augmentation strategy for long-context pre-training in language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1223–1242. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.62. URL https://aclanthology.org/2025.acl-long.62/.

Longyun Wu, Dawei Zhu, Guangxiang Zhao, Zhuocheng Yu, Junfeng Ran, Xiangyu Wong, Lin Sun, and Sujian Li. Longattn: Selecting long-context training data via token-level attention. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19367–19380. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.991. URL https://aclanthology.org/2025.findings-acl.991/.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1192–1199, 2008. doi: 10.1145/1390156.1390306.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4643–4663. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.260. URL https://aclanthology.org/2024.naacl-long.260/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei

Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1M technical report. *arXiv preprint arXiv:2501.15383*, 2025b. URL https://arxiv.org/abs/2501.15383.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025a. URL https://arxiv.org/abs/2506.05176.

Zhiyang Zhang, Ziqiang Liu, Huiming Wang, Renke Shan, Li Kuang, Lu Wang, and De Wen Soh. Re^{3}syn: A dependency-based data synthesis framework for long-context post-training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31468–31480. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.acl-long.1518. URL https://aclanthology.org/2025.acl-long.1518/.

Liang Zhao, Tianwen Wei, Liang Zeng, Cheng Cheng, Liu Yang, Peng Cheng, Lijie Wang, Chenxia Li, Xuejie Wu, Bo Zhu, Yimeng Gan, Rui Hu, Shuicheng Yan, Han Fang, and Yahui Zhou. LongSkywork: A training recipe for efficiently extending context length in large language models. *arXiv preprint arXiv:2406.00605*, 2024. URL https://arxiv.org/abs/2406.00605.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. PoSE: Efficient context window extension of LLMs via positional skip-wise training. In *International Conference on Learning Representations*, volume 2024, pp. 18940–18954, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/524ef58c2bd075775861234266e5e020-Abstract-Conference.html.

Yonghao Zhuang, Lanxiang Hu, Longfei Yun, Souvik Kundu, Zhengzhong Liu, Eric P. Xing, and Hao Zhang. Scaling long context training data by long-distance referrals. In *International Conference on Learning Representations*, volume 2025, pp. 101607–101624, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/fbf49f35b6dd2bef1ad12768c0509daa-Abstract-Conference.html.

A Training Configurations

Tabs. 6 to 9 report the optimization and hardware settings for projector alignment, composer training, long-context continued pretraining, and LongMagpie supervised fine-tuning. Stage 1 ran for 100K optimizer steps with a cosine learning-rate schedule, and Stage 2 was initialized from the resulting 100K projector checkpoint. We use the Stage 2 checkpoint at 78K steps for the main synthesis experiments. All downstream results use the continued-pretraining checkpoint at 500 optimizer steps. In Stage 1, batch sizes count chunks; in Stage 2, they count anchor–candidate training examples.

Table 6: Training configuration for Stage 1 projector alignment.

Parameter	Setting
Frozen LLM	Qwen3-4B
Optimizer	AdamW
β_1, β_2	$0.9, 0.999$
Learning rate	2×10^{-4}
Precision	bfloat16
Gradient clipping	1.0
Weight decay	0
LR schedule	cosine
Optimizer steps	100,000
Warmup	2,000 steps
Global batch size	320
GPU	16× NVIDIA H200

Table 7: Training configuration for Stage 2 composer training.

Parameter	Setting
Initial model	Qwen3-4B
Optimizer	fused AdamW
β_1, β_2	$0.9, 0.95$
Learning rate	5×10^{-5}
Precision	bfloat16
Gradient clipping	1.0
Weight decay	0
LR schedule	cosine
Training duration	100 epochs
Scheduled steps	102,800
Selected checkpoint	78,000 steps
Warmup	1,028 steps (1%)
Max. candidates	1,920 chunks
Max. rank targets	448 chunks
Global batch size	256
GPU	16× NVIDIA H200

Table 8: Training configuration for long-context continued pretraining.

Parameter	Setting
Initial model	Qwen2.5-3B
Optimizer	fused AdamW
β_1, β_2	$0.9, 0.95$
Learning rate	4×10^{-5}
Precision	bfloat16
Gradient clipping	1.0
Weight decay	0.1
LR schedule	cosine
Training steps	500
Warmup	50 steps
Context length	128K
Qwen YaRN factor	4
Global batch size	32
GPU	16× NVIDIA H200

Table 9: Training configuration for LongMagpie supervised fine-tuning.

Parameter	Setting
Packing length	64K
Optimizer	AdamW
β_1, β_2	$0.9, 0.95$
Learning rate	2×10^{-5}
Precision	bfloat16
Gradient clipping	1.0
Weight decay	0.1
LR schedule	cosine
Optimizer steps	500
Warmup	5%
Qwen YaRN factor	4
Global batch size	64
GPU	16× NVIDIA H200

B LongFilter Scoring and Document Selection

We apply LongFilter (Deng et al., 2026) to select positive examples from SlimPajama and Dolma3, using the token-level score in equation (1). Both losses are computed by the same scoring model. For L_i^{long}, the context contains the preceding tokens available within a long-context window of up to 131,072 tokens. For L_i^{short}, it contains only the preceding tokens available within a local 4,096-token window; consecutive short-context windows overlap by 2,048 tokens.

Our setup differs from the original LongFilter experiments in the scoring model, long-context window, and aggregation unit. We use Qwen3.5-4B (Qwen Team, 2026), which natively supports contexts of up to 262,144 tokens, and divide each source document into the consecutive, non-overlapping long-context windows defined above. Rather than assigning a separate score to each fixed-length window, we assign one score to the complete document. Specifically, `score_mean` is the token-weighted mean of all valid token scores across its windows, so a shorter final window contributes in proportion to the number of tokens it contains.

Document selection. We retain the top 30% of documents by `score_mean` separately from SlimPajama and Dolma3, yielding 88.0K and 168.7K documents, respectively. We do not apply LongFilter to the 6.0K The Stack sequences released by ProLong because this collection is comparatively small, and retain all of them for composer training. Only retained documents provide ground-truth continuations. Chunks from both retained and non-retained documents remain in the retrieval corpus and may serve as distractors, except those from the source document of the current training example.

C Neighbor Retrieval and Candidate Construction

ComposeLong does not require a particular retrieval algorithm. It assumes only that a retriever can propose a bounded candidate set. Retrieval limits the search space, but its scores and ranks are not exposed to the composer. We shuffle every candidate pool before composition so that the composer makes its decisions from chunk representations rather than presentation order.

Neighbor retrieval for composer training. For Stage 2, we build a BM25 index over the chunks in the composer-training corpus and precompute a ranked neighbor list for every chunk in each selected real long document. When constructing a training example, we exclude all chunks from its source document. We request 128–1,280 hard negatives, drawing first from neighbors of the anchor and then from neighbors retrieved with other chunks in the same source span. We keep all valid hard negatives when fewer than requested are available. We independently sample 64–192 random negatives from the remaining corpus, remove duplicates, and shuffle them together with the ground-truth continuation. LongFilter determines which documents provide positive continuations but does not restrict the negative pool.

Candidate construction at inference. At inference, we compare document-level BM25 retrieval with chunk-level dense retrieval. The BM25 variant retrieves short documents that are topically related to each source document, following LiteLong (Jia et al., 2026c), and uses their chunks as candidates. The dense variant directly retrieves neighboring chunks using normalized 1,024-dimensional embeddings and inner-product search. We implement it with a FAISS IVF-PQ index containing 131,072 inverted lists and 64 product-quantization subspaces with 8 bits per subspace, and probe 64 lists for each query. Our main experiments use document-level BM25, which performs better in the analysis in Sec. 4.3.

For either retrieval strategy, we split the source document into chunks and process them in their original order. Each source chunk serves as a separate anchor. For each later anchor in the same output, we exclude chunks selected for earlier anchors. We remove duplicate chunks and shuffle each pool before passing it to the composer.

We allocate the target length evenly across the source chunks. For a source document containing p chunks and S characters, the composer selects

$$k = \max\left(0, \left\lceil \frac{T\rho\gamma - S}{p\ell} \right\rceil\right) \tag{4}$$

continuation chunks for each anchor, where $T = 131{,}072$ is the target length in tokens, $\rho = 4.8$ is the estimated number of characters per token, $\gamma = 1.5$ is a margin for variation in chunk lengths, and $\ell = 2{,}048$ is the target chunk length in characters. Decoding runs for this fixed budget rather than predicting a stop symbol. The anchor–continuation groups are then concatenated in source order, and the final length may vary because the chunker preserves natural boundaries rather than producing fixed-length chunks.

D Evaluation Details

Qwen2.5-3B RULER evaluation. For Tab. 2, we use the upstream RULER implementation in `lm-evaluation-harness` (Gao et al., 2023) at commit `d6de816`. It contains 13 tasks: eight needle-in-a-haystack variants, variable tracking, common-words extraction, frequent-words extraction, and two question-answering tasks. We generate 500 examples per task at each of 8K, 16K, 32K, 64K, and 128K using the Qwen tokenizer. All models use the official task templates, zero-shot greedy decoding, and no chat template. We macro-average the 13 task scores at each length and then average the five length scores to obtain *Avg*.

LongBench v2 evaluation. For Tab. 3, we evaluate all models on the same 503 examples using the official LongBench v2 implementation (Bai et al., 2025) at commit `2e00731`. We retain the official zero-shot prompts, answer extraction, and scoring procedure, use each checkpoint's Qwen chat template, and count unparsed answers as incorrect. Inputs longer than 120K Qwen tokens are truncated by retaining the first and last 60K tokens. We decode with temperature 0.1 and a maximum of 128 generated tokens, without chain-of-thought prompting or retrieval augmentation.