

Language Models Improve When Pretraining Data Matches Target Tasks

David Mizrahi¹ Anders Boesen Lindbo Larsen¹ Jesse Allardice¹
 Suzie Petryk¹ Yuri Gorokhov¹ Jeffrey Li^{2†} Alex Fang^{1,3†}
 Josh Gardner[†] Tom Gunter[‡] Afshin Dehghan¹

¹Apple ²University of Washington ³Stanford

Abstract

Every data selection method inherently has a target. In practice, these targets often emerge implicitly through benchmark-driven iteration: researchers develop selection strategies, train models, measure benchmark performance, then refine accordingly. This raises a natural question: what happens when we make this optimization explicit? To explore this, we propose benchmark-targeted ranking (BETR), a simple method that selects pretraining documents based on similarity to benchmark training examples. BETR embeds benchmark examples and a sample of pretraining documents in a shared space, scores this sample by similarity to benchmarks, then trains a lightweight classifier to predict these scores for the full corpus.

We compare data selection methods by training over 500 models spanning 10^{19} to 10^{22} FLOPs and fitting scaling laws to them. From this, we find that simply aligning pretraining data to evaluation benchmarks using BETR achieves a 2.1x compute multiplier over DCLM-Baseline (4.7x over unfiltered data) and improves performance on 9 out of 10 tasks across all scales. BETR also generalizes well: when targeting a diverse set of benchmarks disjoint from our evaluation suite, it still matches or outperforms baselines. Our scaling analysis further reveals a clear trend: larger models require less aggressive filtering. Overall, our findings show that directly matching pretraining data to target tasks precisely shapes model capabilities and highlight that optimal selection strategies must adapt to model scale.

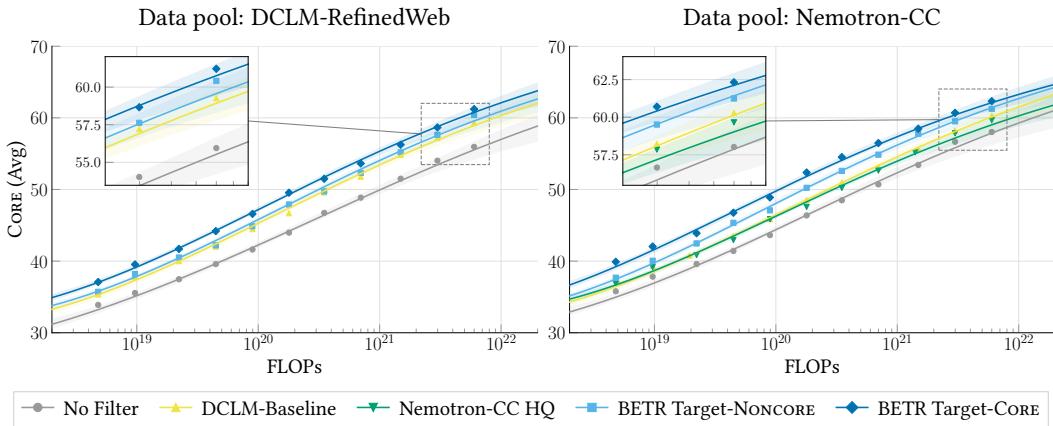


Figure 1: **Benchmark-targeted ranking (BETR) achieves a 1.8x–2.8x compute multiplier over strong baselines.** Scaling curves show accuracy on CORE (10 standard benchmarks) at compute-optimality from 10^{19} to 10^{22} FLOPs. **Target-CORE** directly optimizes for evaluated benchmarks, while **Target-NONCORE** targets distinct benchmarks. Both outperform DCLM-Baseline at all scales.

†Work done during an internship at Apple.

‡Work done while at Apple. Josh Gardner is now at Anthropic.

1 Introduction

Pretraining data fundamentally shapes the capabilities of large language models (LLMs). Data improvements have been a key driver in LLM advances [OpenAI, 2024; Anthropic, 2025; Gemini Team, 2025; DeepSeek-AI, 2024; Meta AI, 2024; Qwen Team, 2025], with recent document-level data selection methods [Penedo et al., 2024; Li et al., 2024; Su et al., 2024a] showing that carefully selected datasets can significantly improve performance across a wide range of tasks. However, current methods largely rely on researcher intuition about what constitutes quality, whether using LLMs to make quality judgments based on implicit criteria [Penedo et al., 2024; Kong et al., 2024] or manually selecting examples of “high-quality” documents for training classifiers, then iterating based on benchmark results [Li et al., 2024]. This makes datasets difficult to systematically improve, as the very notion of “quality” remains loosely defined.

This ambiguity arises because **quality is inherently task-dependent**. Documents valuable for one task often differ from those useful for another. Without specific objectives, ranking diverse documents becomes meaningless: how would one compare the relative quality of a mathematical proof, a poem, and a news article? In practice, however, data selection methods already have implicit objectives: they consistently iterate based on performance measured against standard benchmarks [Li et al., 2024]. This raises a natural question: if benchmark performance implicitly guides the development of data selection methods, what happens when we **explicitly align** pretraining data with these benchmarks?

To explore this, we propose **benchmark-targeted ranking (BETR)**, which directly aligns pretraining data with benchmarks by ranking documents based on their similarity to benchmark examples. BETR operates in three steps: (1) embedding benchmark examples together with a small sample of pretraining documents, (2) scoring documents based on their similarity to benchmark examples, and (3) training an element-wise document scorer to efficiently predict these scores for the full data pool. To ensure evaluation integrity, we target only the training sets of benchmarks and perform careful decontamination against test sets.

We evaluate BETR by training over 500 models spanning compute budgets from 10^{19} to 10^{22} FLOPs, and find that explicitly aligning pretraining data with benchmarks yields consistent improvements, achieving **1.8–2.8x compute multipliers** over state-of-the-art baselines across all scales tested. These results demonstrate that direct data-task matching provides better performance at fixed compute budgets, or equivalently, matches baseline performance using only 35–55% of the compute.

Beyond performance improvements, BETR shows how **benchmark selection fundamentally shapes model capabilities**. The method can create specialist models by targeting individual benchmarks, or generalist models by targeting diverse benchmark sets—each with distinct but complementary strengths. When trained toward standard evaluation benchmarks, models consistently achieve top performance on those tasks, but at the cost of broader capabilities. In contrast, when trained toward an entirely different set of 39 diverse benchmarks, models remain competitive on our evaluation tasks despite never explicitly optimizing for them. This suggests that diverse benchmark targeting fosters broadly useful capabilities rather than narrow, task-specific behaviors.

Our scaling analysis moves beyond the single-scale evaluations common in previous work and reveals a clear trend: **optimal data filtering strategies vary predictably with model scale**. Specifically, smaller models perform best with aggressively filtered data (top 3% at 10^{20} FLOPs), while larger models benefit from greater data diversity (top 30% at 10^{23} FLOPs). The consistency of this relationship across data pools suggests that fixed filtering rates currently used in practice may be suboptimal when applied across varying compute budgets.

While the premise that benchmark-relevant data improves benchmark performance may seem self-evident, its implications (and effective execution) are not. Our work highlights an often-overlooked reality: benchmarks do not merely measure progress, they implicitly guide it. By explicitly aligning data with benchmarks, BETR provides a practical means to improve model training and allows us to understand how benchmark choices shape—and constrain—progress in language modeling.

2 Related work

2.1 Data selection for language models

Web crawl data undergoes multiple preprocessing stages, including HTML parsing, text cleaning, deduplication, and coarse filtering, with the goal to select the parts of the Web that are relevant for pretraining while filtering to an appropriate size for the given compute budget. For a broad coverage of such pipelines, we refer to Gopher [Rae et al., 2021], DCLM [Li et al., 2024], FineWeb [Penedo et al., 2024], and the survey by Albalak et al. [2024]. The data selection methods we discuss operate at the end of this pipeline.

These data selection methods can be organized along two dimensions: **(1) the granularity at which they operate**, from group-level to document-level selection, and **(2) how directly they optimize for target tasks**. In practice, direct optimization becomes less tractable at finer granularities, requiring different strategies at each level.

Group-level selection methods work with grouped data, where groups may be defined by dataset boundaries [Gao et al., 2020], topics and formats [Wettig et al., 2025], web domains [Thrush et al., 2024], or learned clusters [Diao et al., 2025]. These methods aim to find group-level mixture weights that maximize performance on target tasks. The approaches range from manual mixtures based on relative size and expected importance [Brown et al., 2020] to learned methods. Learned approaches can be either offline, where mixtures are tuned through separate experiments [Xie et al., 2023a; Fan et al., 2023; Liu et al., 2024; Grangier et al., 2024; Diao et al., 2025; Shukor et al., 2025], or online, with mixture weights dynamically adjusted during model training [Chen et al., 2023; Albalak et al., 2023; Jiang et al., 2024; Chen et al., 2024b]. Despite their sophisticated optimization procedures, these methods remain constrained, as they treat all documents within a group as equivalent, and the groupings themselves (whether by dataset, domain, format, or topic) are largely heuristic.

Document-level selection methods address the limitations of group-level approaches by scoring at the level of individual documents, but face a different challenge: with billions of documents in web-scale corpora, direct optimization for target tasks becomes computationally intractable. Instead, these methods typically apply element-wise scoring models (e.g. [Fang et al., 2024]) that can efficiently label each document. The scoring approaches vary widely: optimization algorithms that learn scorers for specific objectives [Yu et al., 2024b; Chen et al., 2024c; Brandfonbrener et al., 2024b; Shum et al., 2025; Gu et al., 2025; Calian et al., 2025], perplexity-based filtering [Marion et al., 2023; Ankner et al., 2024] (which has been extended to token-level selection [Lin et al., 2024]), domain matching to either “high-quality” web sources [Xie et al., 2023b] or specific domains [Shao et al., 2024; Zhu et al., 2024], LLM-based quality scoring [Sachdeva et al., 2024; Wettig et al., 2024; Penedo et al., 2024; Kong et al., 2024], classifiers trained to distinguish manually curated “high-quality” examples from general web [Li et al., 2024], and ensemble methods that combine multiple scores and annotations [Su et al., 2024a; Hojel et al., 2025]. While the more sophisticated approaches seem appealing, DCLM showed that a simple FastText classifier [Joulin et al., 2016] outperforms embedding-based methods [Abbas et al., 2023; Xiao et al., 2024], perplexity filtering [Wenzek et al., 2019], and LLM scoring [Sachdeva et al., 2024]. However, finding the right positive examples (Reddit ELI5 [Fan et al., 2019] + OpenHermes-2.5 [Teknium, 2023]) involved iterative experimentation where manually chosen candidates were systematically tested against benchmarks [Li et al., 2024].

Both group- and document-level methods ultimately optimize for benchmark performance, just through different levels of indirection.¹ Group-level methods optimize mixture weights directly but only at coarse granularity, while document-level methods usually rely on proxies refined through benchmark feedback. BETR explores a more direct path by explicitly aligning pretraining documents with benchmark training examples. This allows for document-level selection without the need for iterative refinement or assumptions about what constitutes quality.

¹For an informal discussion on this point, see Beyer [2025].

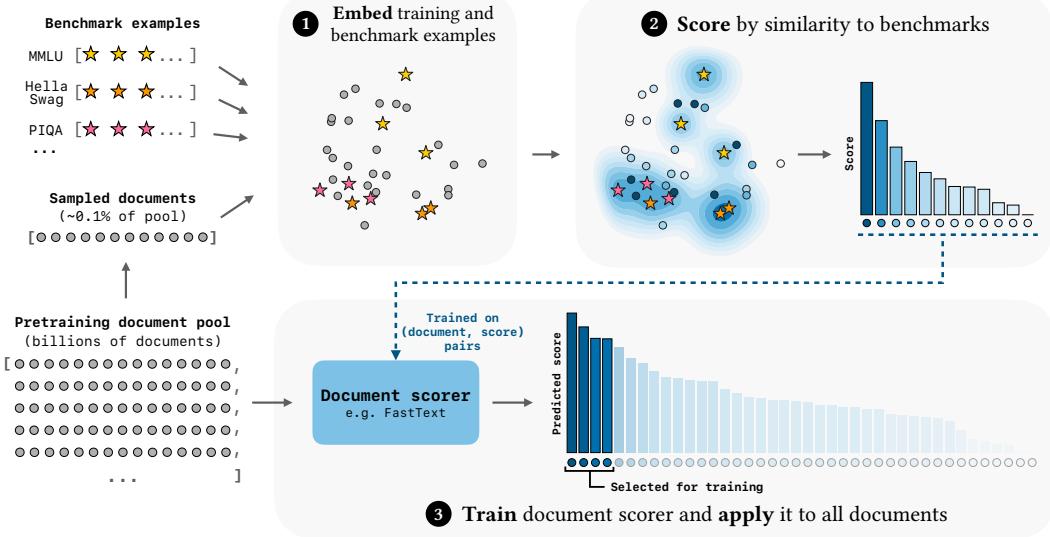


Figure 2: **BETR method overview**. We embed benchmark examples and a small sample of pretraining documents ($\sim 0.1\%$ of pool) in a shared space, score the sampled documents by their similarity to benchmarks, then train a classifier on these scores to efficiently rank and filter the entire document pool.

2.2 Data scaling laws

Scaling laws enable prediction of model behavior across compute scales, initially focused on how loss decreases with model size and training tokens [Kaplan et al., 2020; Hoffmann et al., 2022]. Recent work shows that different datasets exhibit distinct scaling properties: compute-optimality varies by data source [Pandey, 2024; Bi et al., 2024], and losses from different datasets transfer with predictable relationships [Brandfonbrener et al., 2024a; Mayilvahanan et al., 2025]. Special attention has been given to data-constrained regimes where limited data must be repeated, both for language models [Muennighoff et al., 2023] and contrastive vision-language models [Goyal et al., 2024]. For vision-language models, Goyal et al. [2024] found that larger CLIP models benefit from less aggressive filtering when data is constrained. We find that optimal filtering rates for BETR change predictably with scale, extending Goyal et al. [2024]’s observations to language model pretraining and showing this pattern appears even before entering data-constrained regimes (i.e., without multiple training epochs).

Comparing datasets at scale presents unique challenges since pretraining loss alone does not always predict downstream performance [Schaeffer et al., 2024], and because each dataset has its own scaling slope and irreducible loss floor [Pandey, 2024]. Recent methods address this through two-step prediction: first predicting loss from scale, then mapping loss to benchmark accuracy [Gadre et al., 2024; Meta AI, 2024; Bhagia et al., 2024], which we also employ in this work. While DataDecide [Magnusson et al., 2025] compared single-scale and multi-scale evaluation approaches, ultimately advocating for single-scale evaluation, we employ both: single-scale experiments for rapid comparison and ablations, and scaling laws to verify relative rankings across scales and quantify multipliers between methods.

3 Method

3.1 Motivation and problem setting

What happens if we directly align pretraining data with target tasks? To answer this, we need a principled method for selecting pretraining documents that explicitly leverages benchmarks. Such an approach should scale easily to web-scale corpora ($>10B$ documents) and operate at the document-level, rather than at the topic- or domain-level, given the advantages of document-level selection shown by Li et al. [2024]; Penedo et al. [2024]; Su et al. [2024a].

With these goals in mind, we frame data selection as a document ranking problem. Ranking documents requires clearly defined targets, since without targets it is unclear what makes one document better or more relevant than another. Benchmarks provide these targets by representing desired tasks and capabilities. Formally, given a large pool of pretraining documents $\mathcal{D} = \{d_1, \dots, d_N\}$ and a much smaller set of benchmark training examples $\mathcal{B} = \{b_1, \dots, b_M\}$ representing our targets, our goal is to learn a scoring function $s : \mathcal{D} \rightarrow [0, 1]$ that measures each document’s contribution to benchmark performance. These scores enable us to rank and filter the training pool, for example by selecting only the highest-scoring documents for model training.

3.2 Method overview

Given access to both a pretraining data pool and target benchmark examples, benchmark-targeted ranking (BETR) operates in three steps, illustrated in Figure 2. First, we embed both benchmark examples and a sample of pretraining documents in a shared space. Second, we score the sampled documents based on their proximity to benchmark examples. Third, we train a classifier on these scored documents to efficiently predict scores for the entire data pool, and finally use these predicted scores for ranking and filtering.

3.3 Embedding documents and benchmarks

To rank pretraining documents by similarity to benchmarks, we first embed both into a shared space. This step involves three key design choices: how many pretraining documents to sample, how to represent benchmark targets, and which embedding model to use.

Document sampling. While we ultimately need scores for billions of pretraining documents, computing similarities between every pretraining document and every benchmark example is computationally prohibitive. Instead, we score a representative sample directly and later train a classifier to predict these scores for the full corpus (see Section 3.5). This sample size needs to capture the variety of web text without requiring excessive compute for the initial scoring phase.

In practice: We sample 10M documents (less than 0.1% of the total data pool).

Target granularity. We must also decide how to aggregate benchmark examples into embedding targets, which controls the granularity of these targets. Options range from embedding each benchmark example individually (most granular), to computing one centroid per benchmark, to using a single global reference point (least granular).

In practice: We embed each example individually, as we found it performs best empirically (see Section 5.5).

Embedding model. The embedding space must meaningfully capture relationships between web documents and benchmark examples. We use BERT-like transformer encoders [Devlin et al., 2019], as they are simple and widely-used for document retrieval. However, our approach is more general: it can use any method capable of ranking documents relative to a given target, such as rerankers or alternative embedding spaces [Thrush et al., 2024; Shum et al., 2025].

In practice: We use Arctic-Embed L 2.0 [Yu et al., 2024a] for main results and GTE Large v1.5 [Li et al., 2023; Zhang et al., 2024b] for ablations (see Section 5.5).

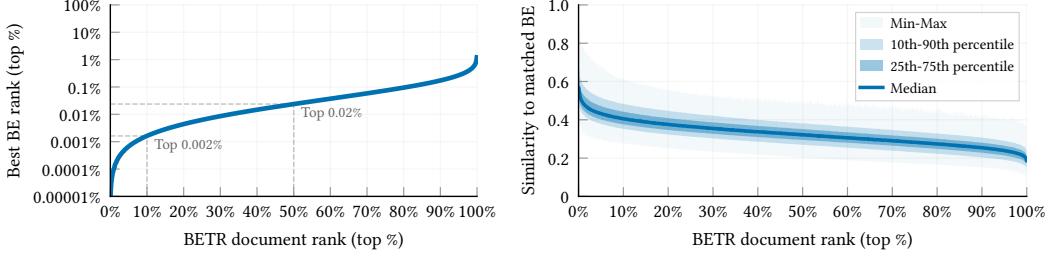


Figure 3: **BETR scoring distributions in practice.** **Left:** Best rank assigned by any benchmark example (BE). With >14,000 benchmark examples competing to score documents, only those ranked in the top 0.002% by some benchmark example reach the top 10% of BETR scores. **Right:** Cosine similarity to the benchmark example that assigned the best rank. Even top-ranked documents show only moderate similarities (~ 0.5) to benchmark examples, highlighting the limited overlap between benchmark examples and web text. Shown for our default (i.e. “*in practice*”) Target-CORE settings on DCLM-RefinedWeb.

3.4 Scoring documents by similarity to benchmarks

Given embeddings for documents and benchmark examples, the next step is to measure how closely each document aligns with these benchmarks. Since each document is compared against thousands of benchmark examples, it is not immediately clear how to aggregate this information into a single, useful document score. Should we reward documents that are extremely close to just one benchmark example, or those that are moderately close to many?

To address this, we define a scoring function based on similarity ranks. Formally, each benchmark example ranks all documents by similarity, giving each document d_j a rank r_{ij} relative to benchmark example i (i.e., how highly benchmark example i ranks document j compared to other documents). We then apply a function $v(r)$ that assigns a value to each rank. These values are aggregated either by averaging, $S_j = \text{mean}_i\{v(r_{ij})\}$, or by taking the maximum value, $S_j = \max_i\{v(r_{ij})\}$.²

The choice of the value function $v(r)$ and aggregation method creates a spectrum of scoring behaviors. Mean aggregation with $v(r) = \log_2(1/r)$ rewards documents with high average relevance across benchmark examples. Max aggregation rewards documents with very high relevance to any single benchmark example. Between these extremes, steeper functions like $v(r) = 1/r$ increasingly favor the best matches.

In practice: We use max aggregation, which scores documents by their best rank across all benchmark examples (see Section 5.5 and Figure 3).

3.5 Predicting scores for the full data pool

As mentioned in Section 3.3, directly computing similarities between all pretraining documents and all benchmark examples is not tractable. Instead, we predict scores for the entire corpus by training an element-wise model (classifier or regressor). We consider different models, such as simple FastText classifiers [Joulin et al., 2016] or fine-tuned language models, which offer tradeoffs between inference speed and prediction accuracy. After scoring, we filter documents by retaining a fixed percentage of tokens (rather than documents) to ensure fair comparisons across methods.

In practice: We train a FastText classifier to predict whether documents are in the top 10% or bottom 90% by rank, downsampling the majority class to balance the training data (see Section 5.5). By default, we retain the top 10% of tokens after scoring and compare different filtering rates in Section 6.2.

²This approach implicitly performs a form of distribution matching. We also explored explicit distribution matching methods (e.g. optimal transport [Peyré et al., 2019]), but found the rank-based approach simpler and easier to scale.

3.6 Targeting strategies

BETR can be applied in two ways, depending on whether the goal is optimizing for specific tasks or general capabilities:

Evaluation-aware (EA). When we know what capabilities we want to optimize for, we target their corresponding benchmarks directly. We use equal numbers of examples from each benchmark to prevent larger benchmarks from dominating the selection and target only benchmark training sets to maintain evaluation integrity.

Evaluation-blind (EB). When we want generally capable models, we target many diverse benchmarks while holding out our evaluation suite. We motivate this approach by observing that benchmarks capture human judgments about useful tasks, so targeting diverse benchmarks should select for broadly valuable text. We ensure no overlap between targeting and evaluation benchmarks and do not iterate based on evaluation results (as this would become evaluation-aware).

4 Experimental setup

4.1 Benchmarks

We define two distinct sets of benchmarks, used for both targeting and evaluation.

CORE benchmarks. This set consists of MMLU [Hendrycks et al., 2020] and 9 benchmarks from the CoreEN evaluation suite [Gunter et al., 2024; Busbridge et al., 2025]: ARC-Easy and ARC-Challenge [Clark et al., 2018], HellaSwag [Zellers et al., 2019], Lambada OpenAI [Paperno et al., 2016]³, PIQA [Bisk et al., 2020], SciQ [Welbl et al., 2017], TriviaQA [Joshi et al., 2017], WebQuestions [Berant et al., 2013], and WinoGrande [Sakaguchi et al., 2021]. These are standard benchmarks that largely overlap with those commonly used in recent data selection work [Li et al., 2024; Penedo et al., 2024; Su et al., 2024a; Thrush et al., 2024; Liu et al., 2024; Kong et al., 2024; Shum et al., 2025]. Targeting CORE train sets and evaluating on CORE test sets corresponds to our evaluation-aware (EA) setting.

NONCORE benchmarks. To test whether benchmark-aligned data generalizes beyond targeted tasks, we construct a diverse benchmark set disjoint from CORE. We take DCLM’s Core and Extended evaluation suite [Li et al., 2024] and remove any benchmarks that overlap with CORE (either fully or partially). This gives us 39 benchmarks (detailed in Appendix B.1). Targeting NONCORE benchmarks while evaluating on CORE corresponds to our evaluation-blind (EB) setting.

4.2 Data pools and baselines

We run our experiments on two data pools. These are large web-crawl corpora with light filtering that serve as the source for our filtering approaches. For each pool, we compare against its best available filtering method.

DCLM-RefinedWeb [Li et al., 2024]. This data pool processes CommonCrawl through the RefinedWeb pipeline [Penedo et al., 2023]. The corpus contains 24T tokens, though many near-duplicates remain due to the lack of global deduplication [Tokpanov et al., 2024; Fang et al., 2024]. Importantly, to ensure evaluation integrity, we apply n-gram decontamination against CORE test sets following GPT-3’s approach [Brown et al., 2020]. We ablate its impact in Section 5.5 and detail the procedure in Appendix B.2.

³Lambada OpenAI only provides a test set. We reserve 1/3 for targeting and evaluate on the remaining 2/3.

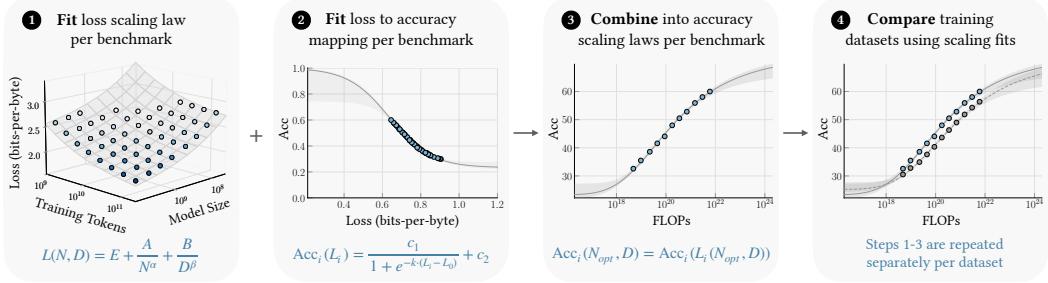


Figure 4: **Comparing datasets using scaling laws.** For each dataset, we use a two-stage scaling law approach to predict per-benchmark accuracy [Gadre et al., 2024; Meta AI, 2024; Bhagia et al., 2024]. 1) Fit a loss scaling law as a function of model size and training tokens (color coding indicates training FLOPs, darker blues for more compute used). 2) Map task loss to accuracy for each benchmark. 3) Combine the two to predict accuracy at any configuration, including compute-optimal settings. 4) Repeat for each dataset to compare performance across compute budgets.

Nemotron-CC [Su et al., 2024a]. This data pool also processes CommonCrawl but with slightly different preprocessing choices and global fuzzy deduplication. Perhaps most importantly, Nemotron-CC includes 1.9T synthetic tokens created through model-based rephrasing⁴ [Maini et al., 2024], resulting in 6.3T total tokens. Since n-gram matching cannot detect rephrased text, we do not decontaminate Nemotron-CC.

Baselines. We compare against two baseline filtering methods. **DCLM-Baseline** [Li et al., 2024] is a FastText classifier trained on Reddit ELI5 + OpenHermes-2.5 as positive examples and RefinedWeb as negatives. When applying it to Nemotron-CC, we retrain the classifier using Nemotron data as negatives instead. For Nemotron-CC, we also compare against **Nemotron-CC HQ** [Su et al., 2024a], which consists of documents classified as high-quality by an ensemble of classifiers⁵. To ensure fair comparison across methods, we filter all datasets to retain the top 10% of tokens, except for Nemotron-CC HQ where we use all high-quality documents (as the quality tiers are discrete).

4.3 Fixed-scale evaluation

We first compare datasets using single-scale experiments [Magnusson et al., 2025; Li et al., 2024], specifically training 7B parameter models for 140B tokens (the “7B-1x” setting in DCLM, roughly compute-optimal per Hoffmann et al. [2022]). This setting represents the smallest scale at which most datasets surpass random performance on MMLU. After validating our approach through ablations at this scale and confirming consistent improvements via scaling laws, we additionally evaluate our main results at the “7B-10x” scale (7B model trained for 1.4T tokens) to test performance in the overtrained regime typical of production models. Further details about our training configuration are provided in Appendix B.3. Since benchmark performance naturally varies, we estimate per-benchmark standard deviation by training 10 models (with different initialization and data shuffling) on a single representative dataset, following T5 [Raffel et al., 2020].⁶

⁴Nemotron-CC rephrases documents selectively: after splitting data into quality tiers, the lowest tier is rephrased once while the highest tier is rephrased in 5 different styles. This effectively increases the proportion of “high-quality” documents in the dataset.

⁵We use all documents labeled as high-quality for Nemotron-CC HQ, while Su et al. [2024a] only use a subset (actual documents and synthetic QA). Appendix E.3 shows their subset performs slightly better, but not enough to change method rankings.

⁶We use variance estimates from the 7B-1x scale for both scales, as estimating them directly at 7B-10x is computationally prohibitive. Actual variance at 7B-10x may differ.

Method	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
Data pool: DCLM-RefinetWeb											
No Filter	77.8	47.7	60.5	74.3	54.6	80.7	94.5	43.8	25.6	73.2	63.3
DCLM-Baseline	81.4	50.7	60.4	75.0	63.1	80.3	95.9	43.4	24.4	73.4	64.8
BETR Target-NONCORE	81.9	51.3	58.3	75.5	63.1	79.4	95.4	46.1	25.0	73.8	65.0
BETR Target-CORE	82.9	53.3	63.1	75.8	62.6	83.2	96.7	47.9	26.0	74.0	66.5
Data pool: Nemotron-CC											
No Filter	82.2	51.5	60.9	71.5	60.8	80.9	95.8	45.3	27.6	72.1	64.9
Nemotron-CC HQ	83.5	54.3	61.1	69.7	67.2	80.8	96.1	45.8	24.7	73.6	65.7
DCLM-Baseline	82.4	53.5	60.8	72.3	67.3	80.4	97.0	44.2	27.5	74.7	65.9
BETR Target-NONCORE	83.4	55.1	58.7	74.0	68.7	79.1	96.1	49.2	26.9	75.2	66.6
BETR Target-CORE	84.0	54.7	64.1	74.9	68.3	83.0	96.4	50.2	25.5	74.0	67.5

Table 1: **CORE results at 7B-10x scale.** We compare BETR variants and baselines at 7B-10x scale (7B parameters, 1.4T tokens). BETR Target-CORE directly optimizes for these benchmarks and achieves the highest average performance. BETR Target-NONCORE targets benchmarks distinct from CORE yet still outperforms baselines, showing generalization beyond targeted tasks. **Bold:** best result ± 1 std.

4.4 Scaling laws evaluation

Single-scale experiments alone cannot determine whether the effects of data selection persist, amplify, or reverse with scale. To address this, we fit scaling laws [Kaplan et al., 2020; Hoffmann et al., 2022] to compare the performance of different data selection methods across varying compute budgets. We train 53 models per dataset, spanning 6×10^{18} to 6×10^{21} FLOPs (>500 models total).

Since training loss cannot be used to meaningfully compare different datasets (easier text yields lower loss regardless of quality), we instead predict benchmark performance [Meta AI, 2024; Gadre et al., 2024; Bhagia et al., 2024]. This is a two-step process: we first model bits-per-byte for each benchmark as a function of model size and training tokens, then map these to per-benchmark accuracy predictions with uncertainty quantification via bootstrapping (Figure 4). This enables us to determine which data selection method achieves higher performance at any compute budget and quantify our confidence in these comparisons.

To summarize the relative efficiency of different data selection methods, we report compute multipliers (CM) [Betker, 2023; Amodei, 2025]. A compute multiplier of X between methods A and B means that method A requires only $1/X$ of the compute to achieve the same performance as method B (at compute-optimality). For example, a 2x compute multiplier indicates that one dataset achieves equivalent performance using half the training compute.

We provide a detailed overview of our scaling law methodology in Appendix C.

5 Results

5.1 Does directly targeting benchmarks improve performance?

We first test whether explicitly aligning pretraining data with target benchmarks improves performance on those same benchmarks. Figure 1 shows that BETR Target-CORE consistently outperforms all baselines on CORE tasks across compute scales from 10^{19} to 10^{22} FLOPs (at compute-optimality). On DCLM-RefinetWeb, BETR achieves a 1.8x compute multiplier over DCLM-Baseline (i.e., requires only 55% of the compute to match DCLM-Baseline performance) and 4.7x over unfiltered data (21%

Target	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
ARC-Easy	81.1¹	49.8 ³	58.4 ⁵	63.2 ⁸	48.1 ⁴	81.0 ⁵	95.5 ²	25.7 ⁴	11.8 ⁶	66.4 ⁸	58.1 ³
ARC-Challenge	80.3 ³	51.4¹	58.1 ⁶	63.3 ⁷	48.4 ³	80.4 ⁶	95.3 ⁴	25.3 ⁶	9.6 ⁹	68.7 ⁶	58.1 ⁴
HellaSwag	74.1 ⁸	40.3 ³	61.5¹	68.2 ⁴	28.7 ⁷	82.0 ²	93.5 ⁷	15.6 ¹¹	7.8 ¹⁰	69.0 ⁵	54.1 ⁸
Lambada	63.7 ¹¹	31.1 ¹¹	53.1 ⁸	75.5²	26.2 ¹¹	74.4 ⁹	89.6 ¹¹	25.5 ⁵	11.7 ⁷	70.6 ³	52.1 ¹¹
MMLU	78.6 ⁵	47.2 ⁵	50.8 ⁹	59.2 ¹⁰	53.0¹	74.1 ¹⁰	95.2 ⁵	25.3 ⁷	15.6 ⁴	69.7 ⁴	56.9 ⁶
PIQA	75.3 ⁷	41.8 ⁷	60.5 ²	65.0 ⁶	27.0 ¹⁰	82.6¹	91.8 ⁸	17.3 ¹⁰	6.9 ¹¹	66.8 ⁷	53.5 ¹⁰
SciQ	80.6 ²	51.2²	57.6 ⁷	57.9 ¹¹	47.6 ⁵	80.1 ⁷	95.9¹	24.8 ⁸	13.3 ⁵	66.2 ⁹	57.5 ⁵
TriviaQA	76.6 ⁶	43.7 ⁶	50.4 ¹⁰	67.0 ⁵	42.2 ⁶	75.1 ⁸	94.2 ⁶	48.8¹	23.1 ²	65.7 ¹⁰	58.7 ²
WebQs	68.8 ¹⁰	34.5 ¹⁰	46.3 ¹¹	63.1 ⁹	28.1 ⁸	71.4 ¹¹	91.7 ⁹	41.8 ²	27.6¹	64.7 ¹¹	53.8 ⁹
WinoGrande	69.2 ⁹	37.0 ⁹	59.8 ⁴	75.7¹	27.7 ⁹	81.1 ⁴	91.4 ¹⁰	21.7 ⁹	10.2 ⁸	72.2¹	54.6 ⁷
All (i.e. CORE)	79.7 ⁴	49.7 ⁴	59.9 ³	74.0 ³	48.9 ²	81.3 ³	95.4 ³	40.4 ³	22.5 ³	71.0 ²	62.3¹

Table 2: **Individual task targeting.** Each row shows performance at 7B-1x scale when targeting a single benchmark (or all jointly). Diagonal cells show performance on the targeted benchmark. Individual targeting excels on the targeted task but often reduces performance on others, while joint targeting balances performance across all tasks. **Bold:** best result ± 1 std. Superscripts indicate rank within column (1 = best).

of compute). On Nemotron-CC, the improvements are even larger: 2.5x over DCLM-Baseline, 2.8x over Nemotron-CC HQ, and 4.7x over no filtering.

These improvements hold across individual benchmarks. BETR Target-CORE outperforms all baselines on 9 out of 10 CORE tasks on both data pools (all except MMLU on DCLM-RefinedWeb, and all except Winogrande on Nemotron-CC). See Appendix Table 5 for per-benchmark compute multipliers.

To verify these improvements persist in the overtrained regime, we evaluate at the 7B-10x scale (6×10^{22} FLOPs). Table 1 shows BETR maintains its advantage on CORE: +1.7 points over DCLM-Baseline on DCLM-RefinedWeb, and +1.8 and +1.6 points over Nemotron-CC HQ and DCLM-Baseline respectively on Nemotron-CC.

Takeaway: Directly targeting benchmarks achieves a 1.8-2.8x compute multiplier over existing methods (4.7x over no filtering), with consistent improvements across scales and tasks.

5.2 Can benchmark targeting precisely control model capabilities?

We next test whether BETR enables fine-grained control over model capabilities by targeting each of the 10 CORE benchmarks individually, then evaluating on all CORE tasks. Table 2 shows clear diagonal dominance: each dataset performs best (or tied best) on its targeted benchmark.

We observe natural clustering within task categories. For example, targeting ARC-Easy improves performance on related knowledge benchmarks like ARC-Challenge, SciQ and MMLU. However, precisely targeting benchmarks in one domain impacts performance in others: e.g., targeting Winogrande improves Lambada (both language understanding benchmarks) but hurts performance on knowledge benchmarks such as MMLU. Targeting all CORE benchmarks jointly (bottom row) avoids these trade-offs, ranking 1st to 4th across all tasks.

Takeaway: Individual benchmark targeting enables precise control but produces specialized models with narrow capabilities.

Target domain	CORE (Avg)
None	56.0
GitHub	52.3
arXiv	52.6
Wikipedia (en)	55.9
StackExchange	56.3
OH-2.5 + ELI5	60.2
DM Mathematics	60.4
All above domains	59.6
NONCORE benchmarks	60.9

Table 3: **Benchmark vs. domain targeting.** Targeting diverse benchmarks (NONCORE) outperforms targeting specific domains on held-out tasks (CORE).

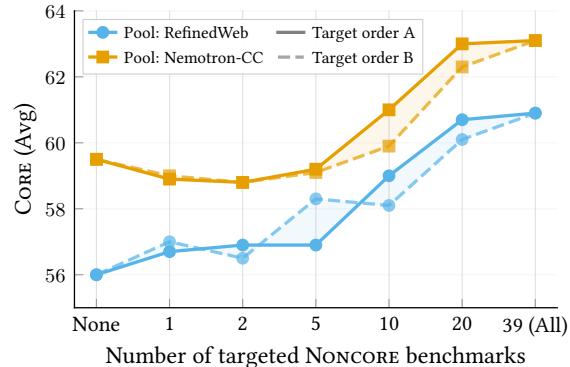


Figure 5: **Performance scales with benchmark diversity.** After an initial plateau, targeting additional diverse benchmarks (NONCORE) consistently improves performance on held-out tasks (CORE).

5.3 Does targeting diverse benchmarks generalize to held-out tasks?

While individual benchmark targeting creates specialized models, we now investigate whether targeting diverse benchmarks can create broadly capable models. We target all 39 NONCORE benchmarks (none overlap with CORE) and evaluate on CORE tasks. This corresponds to our evaluation-blind (EB) setting described in Section 3.6.

We find that BETR Target-NONCORE achieves strong performance on held-out CORE tasks, matching or exceeding baselines across both scaling laws (Figure 1) and at the 7B-10x scale (Table 1). As expected, Target-NONCORE underperforms Target-CORE (which directly targets the evaluation distribution).

To understand why NONCORE-targeting works, we use BETR to target “high-quality” domains instead (Table 3). We test domains from the Pile [Gao et al., 2020] and the OH-2.5 + ELI5 dataset used as positive examples for DCLM-Baseline’s classifier [Li et al., 2024]. Performance varies widely depending on the domain: GitHub (52.3) performs worse than no filtering, while DM Mathematics [Saxton et al., 2019] (60.4) and OH-2.5 + ELI5 (60.2) approach but never reach Target-NONCORE performance (60.9).

We then investigate how performance scales with benchmark diversity (Figure 5). Using two random orderings of the 39 NONCORE benchmarks⁷, we measure performance when targeting an increasing number of benchmarks. Performance stagnates from 1 to 5 benchmarks, then reliably improves as more benchmarks are added. This pattern holds across both orderings and data pools.

Takeaway: Targeting diverse benchmarks yields generally capable models that perform well even on held-out tasks, with performance improving as more benchmarks are targeted.

5.4 What are the consequences of optimizing for a limited set of benchmarks?

Having shown that diverse benchmark targeting creates broadly capable models, we now test whether optimizing for common evaluation benchmarks maintains general capabilities. We evaluate Target-CORE models on held-out NONCORE tasks.

Figure 6 reveals a clear rank reversal. While Target-CORE achieves the best CORE performance (66.5

⁷See Appendix Table 11 for the specific random orderings.

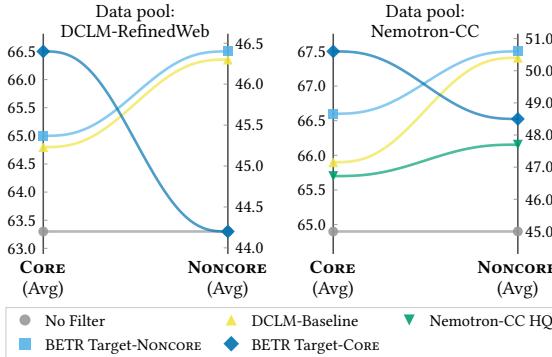


Figure 6: CORE vs NONCORE performance. At 7B-10x scale, BETR Target-CORE achieves the highest CORE performance but falls to third place on held-out NONCORE tasks, showing the trade-off of targeted optimization.

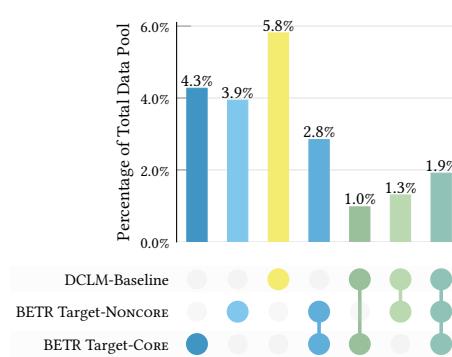


Figure 7: Document overlap analysis. Top 10% document agreement on DCLM-RefinedWeb poorly correlates with model capabilities: similar-performing methods show different selection patterns.

on DCLM-RefinedWeb), it ranks third on NONCORE tasks (44.2), trailing both Target-NONCORE⁸ (46.4) and DCLM-Baseline (46.3). This pattern holds across both data pools.

We analyze document overlap to understand these performance differences (Figure 7). While methods show moderate agreement (the three methods share 19% of their top 10% documents, with pairwise Spearman correlations ranging from 0.44 to 0.71), there is no convergence toward universal “high-quality” data. Furthermore, this agreement poorly predicts capability profiles: Target-NONCORE and DCLM-Baseline achieve nearly identical performance on both CORE and NONCORE tasks despite only 32% agreement, whereas the two BETR variants have 47% agreement yet produce models with different capabilities.

These findings exemplify Goodhart’s Law: despite CORE spanning diverse tasks including world knowledge, commonsense reasoning, and language understanding, these benchmarks cease to be good proxies for general capability when directly optimized. The results show that models improve on exactly what their training data targets, with clear trade-offs on other capabilities. This concern extends beyond our work, as many data selection methods evaluate on fewer benchmarks, with some optimizing for as few as 1–2 evaluation tasks [Thrush et al., 2024; Wettig et al., 2025].

Takeaway: Standard pretraining benchmarks (CORE) represent a narrow slice of capabilities: models optimized exclusively for them underperform on held-out tasks.

5.5 What factors drive performance?

In Figure 8, we systematically ablate each BETR component to understand what drives its effectiveness.

Target granularity strongly affects performance (Figure 8a). Per-example targeting achieves 62.2 CORE accuracy, outperforming k-means clustering (60.3–60.6), per-benchmark averaging (59.5), and global averaging (58.3).

Embedding choice has minimal impact (Figure 8b). Performance is mostly consistent across models, with Arctic-Embed L v2 [Yu et al., 2024a] performing slightly better (62.5)⁹.

⁸Target-NONCORE uses all available benchmark examples without train/test splits or per-benchmark balancing, unlike Target-CORE. Our ablations show that train vs test targeting splits has minimal impact (Table 4).

⁹We also tested applying a reranker (BGE-v2-m3 [Chen et al., 2024a]) to the top 1000 documents per benchmark, which provided no additional benefit.

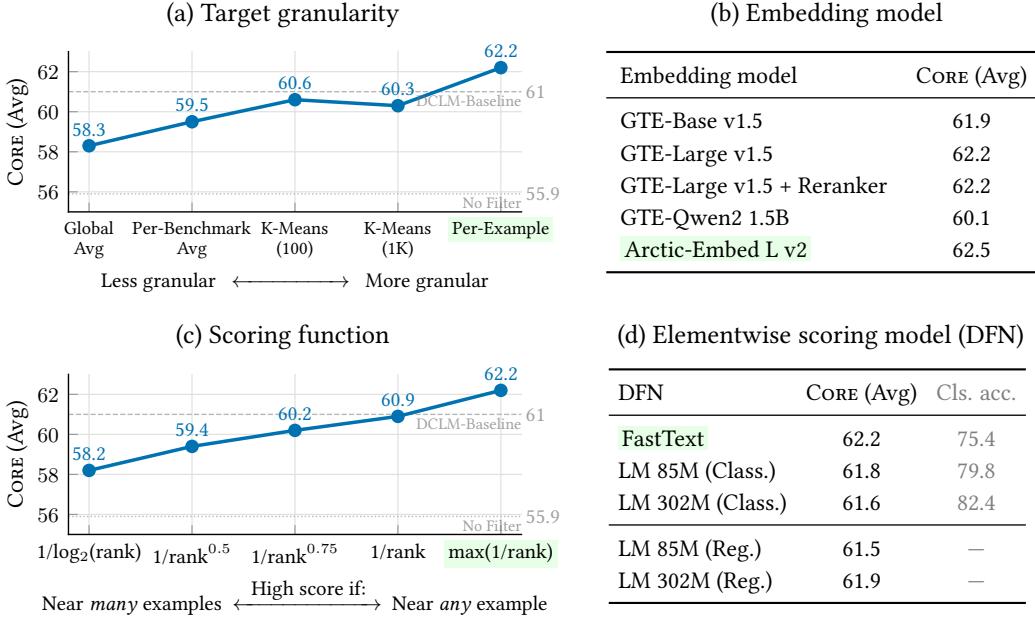


Figure 8: **BETR ablation studies** showing the impact of (a) target granularity, (b) embedding model choice, (c) scoring function, and (d) elementwise scorer on model performance. All experiments use DCLM-RefinedWeb at 7B-1x scale. Highlighted cells indicate selected settings for our main BETR models in Figure 1 and Table 1.

Benchmark target	CORE (Avg)	Decont.	CORE (Avg)	Method	CORE (Avg)	Cls. acc.
Test set (i.e. eval)	62.4	✗	62.5	CORE vs crawl	59.7	99.5
Train set (i.e. non-eval)	62.3	✓	62.3	BETR Target-CORE	62.3	75.6

Table 4: **Method integrity ablations** showing (left) targeting train vs test splits yields similar results, (middle) decontamination has minimal impact, and (right) BETR significantly outperforms directly training a FastText classifier on benchmarks against web crawl.

The scoring function significantly affects results (Figure 8c). Scoring functions that reward documents highly relevant to any single benchmark example (62.2) outperform functions that reward average relevance across multiple examples (58.2).

The choice of classifier has a counterintuitive effect (Figure 8d). A simple FastText classifier matches or exceeds LM-based classifiers and regressors despite lower validation accuracy (75.4% vs 82.4% for 302M LM), which suggests that classification accuracy is not the key metric for effective data selection.

To verify BETR’s integrity, we test whether gains come from test set information or contamination (Table 4). Targeting benchmark train sets performs identically to targeting test sets (62.3 vs 62.4), and decontamination has minimal impact (-0.2), confirming the method works without test set leakage.¹⁰ Finally, directly classifying benchmark vs web text underperforms BETR (59.7 vs 62.2), validating our two-step approach.¹¹

Takeaway: Simple design choices drive BETR’s success: fine-grained targets, scoring functions that favor single-best matches, and simple classifiers outperform more complex alternatives.

¹⁰For an illustration of what happens when evaluation integrity is *not* maintained, see Schaeffer [2023].

¹¹Distinguishing benchmark examples from web text is trivial (99.5% validation accuracy), while BETR’s task of classifying top 10% vs bottom 90% of ranked documents is much harder (75.6% accuracy). We hypothesize that this harder task leads to stronger features for data selection.

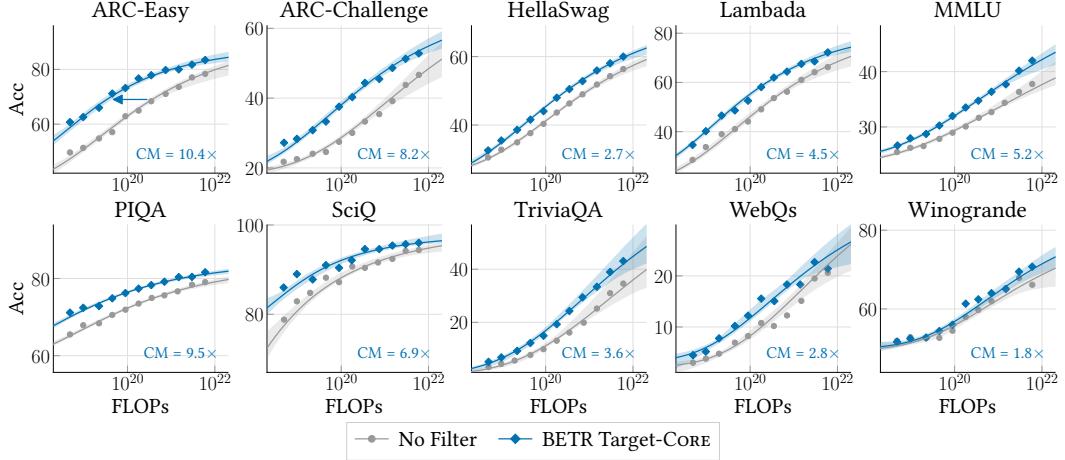


Figure 9: **Per-benchmark scaling laws.** We compare BETR Target-CORE against no filtering across all CORE benchmarks on Nemotron-CC, showing compute-optimal performance from 10^{19} to 10^{22} FLOPs. The scaling laws use two-stage fitting: first from model size and training tokens to per-benchmark bits-per-byte, then from bits-per-byte to accuracy. Compute multipliers (CM) vary by over 5x: from 10.4x for ARC-Easy to 1.8x for Winogrande.

6 Insights from scaling laws

Our scaling analysis reveals additional insights about data selection that only become visible across multiple compute scales. While Section 5 showed that BETR consistently improves CORE performance, examining individual benchmarks and filtering rates reveals important nuances: not all tasks benefit equally from data selection, and optimal filtering strategies must adapt to model scale.

6.1 Tasks vary in how much they benefit from data selection

We test whether all benchmarks benefit equally from data selection. While BETR achieves a 4.7x average compute multiplier over no filtering, examining individual benchmarks reveals whether some tasks are easier to improve through data selection.

Figure 9 shows per-benchmark FLOPs-to-accuracy curves for BETR Target-CORE and no filtering on Nemotron-CC. From these curves, we find substantial variation in the effectiveness of data selection: BETR achieves a 10.4x compute multiplier over no filtering on ARC-Easy and 9.5x on PIQA (i.e., requires 10% of the FLOPs to match no filtering performance), but only 2.8x on WebQuestions and 1.8x on Winogrande. MMLU falls in the middle at 5.2x. See Appendix Table 5 for complete results.

This 6-fold range shows that data selection benefits tasks unevenly, and suggests matching works best when benchmark-relevant information is concentrated in specific documents (e.g., knowledge-intensive benchmarks) rather than distributed across general text (e.g., language understanding benchmarks). All tasks improve with data selection, but some see order-of-magnitude efficiency gains while others benefit primarily from scale.

Takeaway: Data selection benefits tasks unevenly. For BETR Target-CORE, knowledge tasks achieve up to 10x compute multipliers (ARC-Easy) while language understanding tasks show more modest 2x multipliers (Winogrande).

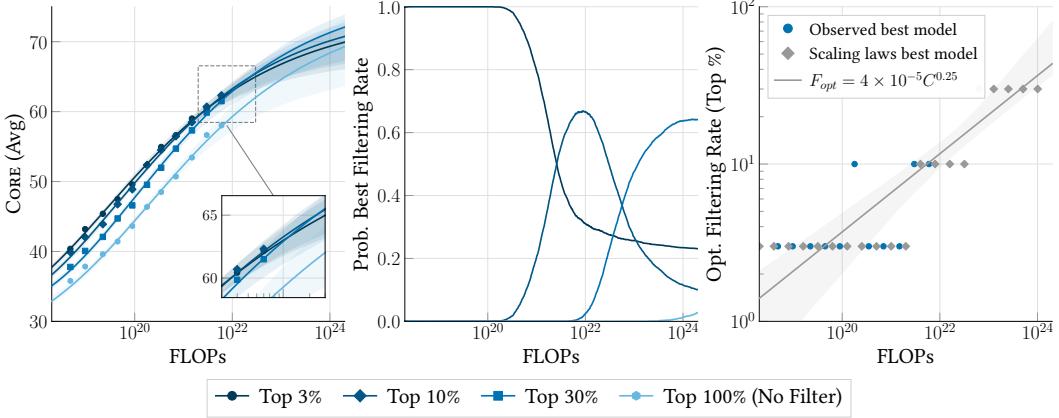


Figure 10: **Optimal data filtering rate scaling.** **Left:** Compute-optimal CORE accuracy for varying BETR Target-CORE filtering rates on Nemotron-CC, along with scaling law fits. **Center:** Probability that each filtering rate is optimal at a given compute scale, estimated from bootstrap distributions of scaling law fits. **Right:** Optimal filtering rate (top- $x\%$) as a function of compute, shown both for the [best observed model](#) and the best model predicted via scaling laws. The scaling law fit $F_{\text{opt}}(C) = 4 \times 10^{-5} C^{0.25}$ is overlaid for comparison.

6.2 Optimal filtering rate depends strongly on scale

Throughout our experiments, each method retains the top 10% of tokens to ensure fair comparisons. However, BETR produces continuous scores that can be used to rank and filter documents at any threshold. We therefore test whether 10% is optimal across scales by training models with BETR Target-CORE on Nemotron-CC at four filtering rates: top 3%, 10%, 30%, and 100% (no filtering).

Figure 10 reveals a clear trend across scales. The left panel shows CORE average accuracy curves for each filtering rate: aggressive top 3% filtering achieves the highest accuracy at small compute budgets but is overtaken by progressively lighter filtering as compute increases. The middle panel quantifies these transitions using bootstrap distributions from our scaling fits. The probability of each rate being optimal shifts smoothly from 3% (below 10^{21} FLOPs) to 10% (around 10^{22} FLOPs) to 30% (beyond 10^{23} FLOPs). The right panel shows this progression follows a power law: $F_{\text{opt}}(C) = 4 \times 10^{-5} \times C^{0.25}$, where F_{opt} is the percentage of tokens retained and C is training FLOPs.¹² These results hold at compute-optimal training. However, when training fixed-size models for increasing token counts (Appendix D.5), the shift toward lighter filtering is much more gradual.

While Goyal et al. [2024] observed similar scale-dependent filtering for CLIP training, their results emerge in data-constrained settings with repeated epochs. Our findings show this pattern holds even in single-epoch language model training, suggesting smaller models inherently require more selective filtering while larger models can extract signal from noisier but more diverse data. Similar patterns appear on DCLM-RefinetWeb (Appendix D.4), indicating this is not data pool-specific.

Takeaway: Optimal filtering predictably becomes less strict as model scale increases. For BETR Target-CORE on Nemotron-CC, the optimal filtering rate (top- $x\%$) follows $F_{\text{opt}} \propto C^{0.25}$, increasing from 3% at 10^{20} FLOPs to 30% at 10^{23} FLOPs.

¹²We fit a power law for its simplicity and good fit in our observed range. Other functional forms (e.g., sigmoid) might better capture behavior at extreme scales where filtering approaches 100%, but we do not reach such scales in our experiments.

7 Discussion & Conclusion

We tested whether language models improve when pretraining data matches target tasks. This hypothesis seems almost self-evident: training on relevant data should naturally improve relevant capabilities. Yet, quantifying this effect and understanding its implications provides important insights into data selection and evaluation practices. When targeting evaluated benchmarks, BETR achieves 1.8–2.8x compute multipliers over existing methods, with consistent improvements from 10^{19} to 10^{23} FLOPs. Additionally, the method works equally well for creating specialists (via individual benchmark targeting) or generalists (via diverse benchmark targeting), making it a valuable tool for explicit data selection.

Beyond its practical utility, BETR also sheds light on important properties of benchmark optimization. Target-CORE achieves the best CORE performance but falls below baselines on held-out NONCORE tasks. In contrast, Target-NONCORE performs best on NONCORE while maintaining competitive CORE performance. This asymmetry shows BETR works as intended: each variant excels where targeted. It also highlights that CORE benchmarks represent a narrower slice of capabilities than their diversity (world knowledge, language understanding, commonsense reasoning) would suggest. This pattern extends beyond BETR: existing methods implicitly optimize for similar benchmarks through iterative development, with BETR simply accelerating this process to reveal its endpoint.¹³ Achieving broader capabilities thus requires targeting benchmarks beyond those commonly used for evaluation.

Although explicit targeting might seem at odds with pretraining’s traditional emphasis on generality, our scaling analysis offers a reconciling insight: as compute increases, optimal filtering becomes predictably less strict. Smaller models perform best when trained on narrowly filtered datasets, while larger models benefit from more diverse data. At sufficient scales, lighter filtering combined with diverse targeting could resolve the specialization/generality trade-off.

Our findings have several limitations. We experiment only with English text, excluding multilingual and code data, where BETR may behave differently.¹⁴ We do not verify if pretraining improvements persist after post-training.¹⁵ Our scaling analysis assumes unlimited data, while aggressive filtering realistically leads to data constraints [Muennighoff et al., 2023].¹⁶ We also focus exclusively on hard filtering at fixed thresholds, rather than softer weighting or multi-stage approaches [Blakeney et al., 2024]. Finally, FastText’s strong performance despite lower classification accuracy than LM alternatives remains unexplained, indicating gaps in our understanding of data selection.

Taken together, our findings demonstrate that explicit data-task matching is both feasible and useful, achieving 2–3x compute multipliers over existing methods while enabling precise control over capabilities. By making data selection targets explicit, BETR reveals that benchmark optimization (whether deliberate or implicit) shapes model capabilities in predictable ways. These findings suggest that progress in language modeling requires not just better data selection methods, but clarity about what capabilities we want and how we measure them.

Acknowledgments. We would like to thank Roman Bachmann, Sam Dodge, Alaaeldin El-Nouby, Doug Kang, Oğuzhan Fatih Kar, Xiang Kong, Justin Lazarow, Brandon McKinzie, Futang Peng, Henry Tran, and Yinfei Yang for helpful feedback and support at various stages of the project.

¹³This reflects Goodhart’s Law: when a measure becomes a target, it ceases to be reliable. CORE benchmarks, initially proxies for general language capabilities, lose their indicative power once explicitly optimized.

¹⁴Applying BETR to multilingual or code data would likely require replacing FastText’s whitespace tokenization with a BPE tokenizer [Zhu et al., 2024].

¹⁵Li et al. [2024] suggest such improvements typically persist.

¹⁶For instance, top 10% filtering of Nemotron-CC yields only 630B tokens, far below the 15T+ tokens typical in production models [Meta AI, 2024; DeepSeek-AI, 2024; Qwen Team, 2025].

References

- Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.
- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebr'on, and Sumit K. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.
- Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.
- Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muenighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.
- Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.
- Dario Amodei. On deepseek and export controls, January 2025. URL <https://www.darioamodei.com/post/on-deepseek-and-export-controls>.
- Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. *arXiv preprint arXiv:2405.20541*, 2024.
- Anthropic. System card: Claude opus 4 & claude sonnet 4. <https://www.anthropic.com/clause-4-system-card>, May 2025. Claude 4 family system card.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL <https://aclanthology.org/D13-1160>.
- Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt. *arXiv preprint arXiv:2404.10102*, 2024.
- James Betker. Compute multipliers, 2023. URL <https://nonint.com/2023/11/05/compute-multipliers/>. Blog post.
- Lucas Beyer. When you think it through, there are only 2 fundamental approaches to data selection. Tweet, March 2025. URL <https://x.com/giffmana/status/1898664177452953701>. Twitter/X.
- Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A Smith, Dirk Groeneveld, Pang Wei Koh, et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.
- Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.
- Cody Blakeney, Mansheej Paul, Brett W Larsen, Sean Owen, and Jonathan Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training. *arXiv preprint arXiv:2406.03476*, 2024.
- David Brandfonbrener, Nikhil Anand, Nikhil Vyas, Eran Malach, and Sham Kakade. Loss-to-loss prediction: Scaling laws for all datasets. *arXiv preprint arXiv:2411.12925*, 2024a.

David Brandfonbrener, Hanlin Zhang, Andreas Kirsch, Jonathan Richard Schwarz, and Sham Kakade. Color-filter: Conditional loss reduction filtering for targeted language model pre-training. *Advances in Neural Information Processing Systems*, 37:97618–97649, 2024b.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Eta Littwin, and Russ Webb. Distillation scaling laws. *arXiv preprint arXiv:2502.08606*, 2025.

Dan A Calian, Gregory Farquhar, Iurii Kemaev, Luisa M Zintgraf, Matteo Hessel, Jeremy Shar, Junhyuk Oh, András György, Tom Schaul, Jeffrey Dean, et al. Datarater: Meta-learned dataset curation. *arXiv preprint arXiv:2505.17895*, 2025.

Jianly Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024a.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36:36000–36040, 2023.

Mayee F Chen, Michael Y Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. *arXiv preprint arXiv:2411.05735*, 2024b.

Xuxi Chen, Zhendong Wang, Daouda Sow, Junjie Yang, Tianlong Chen, Yingbin Liang, Mingyuan Zhou, and Zhangyang Wang. Take the bull by the horns: Hard sample-reweighted continual training improves llm generalization. *arXiv preprint arXiv:2402.14270*, 2024c.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, et al. Climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. *arXiv preprint arXiv:2504.13161*, 2025.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*, 2019.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393*, 2023.

Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander T Toshev, and Vaishaal Shankar. Data filtering networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Sachin Goyal, Pratyush Maini, Zachary C Lipton, Aditi Raghunathan, and J Zico Kolter. Scaling laws for data filtering–data curation cannot be compute agnostic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22702–22711, 2024.

David Grangier, Simin Fan, Skyler Seto, and Pierre Ablin. Task-adaptive pretrained language models via clustered-importance sampling. *arXiv preprint arXiv:2410.03735*, 2024.

Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=dhAL5fy8wS>.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Andrew Hojel, Michael Pust, Tim Romanski, Yash Vanjani, Ritvik Kapila, Mohit Parmar, Adarsh Chaluvaraju, Alok Tripathy, Anil Thomas, Ashish Tanwer, et al. Essential-web v1. 0: 24t tokens of organized web data. *arXiv preprint arXiv:2506.14111*, 2025.

Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Shrimai Prabhumoye, Ayush Dattagupta, Vibhu Jawa, Jiwei Liu, Ryan Wolf, Sarah Yurick, and Varun Singh. NeMo-Curator: a toolkit for data curation. <https://github.com/NVIDIA/NeMo-Curator>, 2025.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *arXiv preprint arXiv:2410.11820*, 2024.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

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

Xiang Kong, Tom Gunter, and Ruoming Pang. Large language model-guided document selection. *arXiv preprint arXiv:2406.04638*, 2024.

Mark Lee, Tom Gunter, Chang Lan, John Peebles, Hanzhi Zhou, Kelvin Zou, Sneha Bangalore, Chung-Cheng Chiu, Nan Du, Xianzhi Du, et al. Axlearn: Modular large model training on heterogeneous infrastructure. *arXiv preprint arXiv:2507.05411*, 2025.

- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.
- Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.
- Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*, 2024.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.
- Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.
- Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, et al. Datadecide: How to predict best pretraining data with small experiments. *arXiv preprint arXiv:2504.11393*, 2025.
- Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint arXiv:2401.16380*, 2024.
- Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.
- Prasanna Mayilvahanan, Thaddäus Wiedemer, Sayak Mallick, Matthias Bethge, and Wieland Brendel. Llms on the line: Data determines loss-to-loss scaling laws. *arXiv preprint arXiv:2502.12120*, 2025.
- Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.
- Meta AI. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
- MosaicML. lilm-foundry: Eval_gauntlet.md. https://github.com/mosaicml/lilm-foundry/blob/main/scripts/eval/local_data/EVAL_GAUNTLET.md, 2023.
- Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.
- OpenAI. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- Rohan Pandey. gzip predicts data-dependent scaling laws. *arXiv preprint arXiv:2405.16684*, 2024.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.
- Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R Bowman. Bbq: A hand-built bias benchmark for question answering. *arXiv preprint arXiv:2110.08193*, 2021.
- Arkil Patel, Satwik Bhattacharya, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

PatronusAI. Patronus ai launches enterprisepii, the industry's first llm dataset for detecting business-sensitive information. <https://www.patronus.ai/announcements/patronus-ai-launches-enterprisepii-the-industrys-first-llm-dataset-for-detecting-business-sensitive-information>, 2023.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Guilherme Penedo, Hynek Kydliček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium*, 2011. URL <https://people.ict.usc.edu/~gordon/copa.html>.

Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. How to train data-efficient llms. *arXiv preprint arXiv:2402.09668*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Rylan Schaeffer. Pretraining on the test set is all you need. *arXiv preprint arXiv:2309.08632*, 2023.

Rylan Schaeffer, Hailey Schoelkopf, Brando Miranda, Gabriel Mukobi, Varun Madan, Adam Ibrahim, Herbie Bradley, Stella Biderman, and Sanmi Koyejo. Why has predicting downstream capabilities of frontier ai models with scale remained elusive? *arXiv preprint arXiv:2406.04391*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noam M. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

- Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *arXiv preprint arXiv:2507.09404*, 2025.
- Kashun Shum, Yuzhen Huang, Hongjian Zou, Qi Ding, Yixuan Liao, Xiaoxin Chen, Qian Liu, and Junxian He. Predictive data selection: The data that predicts is the data that teaches. *arXiv preprint arXiv:2503.00808*, 2025.
- Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adri Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.
- Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*, 2024a.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024b.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.
- Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL <https://huggingface.co/datasets/teknium/OpenHermes-2.5>.
- Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations. *arXiv preprint arXiv:2409.05816*, 2024.
- Yury Tokpanov, Paolo Glorioso, Quentin Anthony, and Beren Millidge. Zyda-2: a 5 trillion token high-quality dataset. *arXiv preprint arXiv:2411.06068*, 2024.
- Bojan Tunguz. 200,000+ jeopardy! questions. Kaggle, 2019. URL <https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.
- Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.
- Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*, 2019.
- Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. QuRating: Selecting high-quality data for training language models. In *International Conference on Machine Learning (ICML)*, 2024.
- Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. *arXiv preprint arXiv:2502.10341*, 2025.
- Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.
- Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649, 2024.
- Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023a.

- Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36:34201–34227, 2023b.
- Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.
- Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*, 2024a.
- Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models. *Advances in Neural Information Processing Systems*, 37:108735–108759, 2024b.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.
- Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How does critical batch size scale in pre-training? *arXiv preprint arXiv:2410.21676*, 2024a.
- Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*, 2024b.
- Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.
- Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.

Appendix

Table of Contents

A Method details	25
B Experimental setup details	25
B.1 Benchmark details	25
B.2 Decontamination	25
B.3 Fixed-scale architecture	26
C Scaling law methodology	26
C.1 Scaling experiments setup	26
C.2 Batch size selection	27
C.3 Loss scaling with model size and training tokens	28
C.4 Task accuracy scaling	31
C.5 CORE average accuracy computation	32
C.6 Compute multipliers	32
D Additional scaling law findings	33
D.1 Compute-optimal token-to-parameter ratios	33
D.2 Per-benchmark scaling laws	34
D.3 Data pool comparison	34
D.4 Optimal filtering rates on DCLM-RefinedWeb	35
D.5 Fixed model size scaling	36
D.6 Example scaling law parameters	37
E Additional experiments & analysis	38
E.1 Detailed NONCORE results	38
E.2 7B-1x CORE results	38
E.3 Nemotron-CC HQ dataset definition	39
E.4 Per-benchmark standard deviations	39
E.5 Benchmark contributions to document selection	40
E.6 Topic and format distributions	41
F Full tables	41

A Method details

We briefly describe additional implementation details for BETR.

Embedding documents. Pretraining documents are embedded directly without any preprocessing. For benchmark examples, we convert each example into a single document by concatenating the question, answer, and any additional information (e.g., SciQ [Welbl et al., 2017] includes a “support” field with explanatory text for the correct answer). The maximum context length is set to 8192 tokens for both Arctic-Embed L v2 [Yu et al., 2024a] and GTE models [Li et al., 2023; Zhang et al., 2024b].

FastText scorer training. All FastText models [Joulin et al., 2016] use whitespace tokenization and 2-grams with the following hyperparameters: learning rate of 0.03, dimension of 128, window size of 10, minimum occurrence count of 5, and 5 training epochs. We selected these values through 1000 Optuna trials [Akiba et al., 2019] on a representative baseline configuration, observing that validation accuracy remained stable across a wide range of hyperparameters near these values.

Language model scorer training. For the LM-based classification and regression scorers [Kong et al., 2024] in Figure 8, we finetune pretrained language models with either a binary classification or regression head. We select hyperparameters through ~ 50 runs, searching over training steps, learning rates, schedules, and whether to freeze the base model.

Filtering thresholds. We determine filtering thresholds by applying scorers to a held-out set of 100K documents and selecting thresholds that retain the desired percentage of tokens (typically 10%). We find that this sample size is sufficient, as thresholds stabilize at 100K documents.

B Experimental setup details

B.1 Benchmark details

CORE evaluation settings. Our CORE evaluation settings follow from Gunter et al. [2024]. ARC-Easy, ARC-Challenge [Clark et al., 2018], HellaSwag [Zellers et al., 2019], Lambada OpenAI [Paperno et al., 2016], PIQA [Bisk et al., 2020], SciQ [Welbl et al., 2017], WinoGrande [Sakaguchi et al., 2021] are 0-shot tasks. TriviaQA [Joshi et al., 2017] and WebQuestions [Berant et al., 2013] are 1-shot tasks. MMLU [Hendrycks et al., 2020] is a 5-shot task.

NONCORE evaluation settings. Our NONCORE evaluation tasks are sourced from the DCLM-Core and DCLM-Extended evaluation sets [Li et al., 2024], taking all tasks that do not (partially) overlap with CORE tasks. The evaluation procedure also follows from Li et al. [2024]. The benchmarks are all listed in Table 11, along with their subcategory from Table 6 and random order from Figure 5.

B.2 Decontamination

We decontaminate DCLM-RefinedWeb using the procedure from GPT-3 [Brown et al., 2020] and MT-NLG [Smith et al., 2022], as implemented in NeMo-Curator [Jennings et al., 2025]. This procedure uses n-gram matching (with n ranging from 8 to 13) to identify overlaps between pretraining data and CORE benchmarks. When a matching n-gram is found, up to 200 characters on each side are removed and the document is split at that point. Documents that get split more than 10 times are discarded entirely. To preserve common phrases, we skip decontamination for n-grams appearing more than 10,000 times in the pretraining dataset. This provides a large safety margin, as we manually observe that the transition between benchmark content and common phrases is around 300 occurrences.

B.3 Fixed-scale architecture

The model used for fixed-scale experiments (7B-1x and 7B-10x scale) is a 6.7B Transformer decoder [Vaswani et al., 2017]. The architecture follows PaLM [Chowdhery et al., 2023] and Llama3 [Meta AI, 2024], i.e., it uses SwiGLU [Shazeer, 2020], GQA [Ainslie et al., 2023], RMSNorm [Zhang and Sennrich, 2019], and RoPE [Su et al., 2024b]. We also use μ P-simple [Yang et al., 2022; Wortsman et al., 2023] and QKNorm [Wortsman et al., 2023]. The model has $n_{\text{layers}} = 32$, $d_{\text{model}} = 4096$, $d_{\text{ff}} = 13056$, $n_{\text{query}} = 32$, $n_{\text{kv}} = 8$, and a context length of 4096. We use AXLearn [Lee et al., 2025] as our training framework.

C Scaling law methodology

This section of the appendix provides complete details of our scaling law methodology, which enables us to compare data selection methods across compute scales and derive the compute multipliers reported in Sections 5 and 6.

C.1 Scaling experiments setup

Overview. To robustly compare how different data selection methods perform across scales, we trained over 500 models spanning 11 different datasets (i.e. data pool + selection method). This large-scale experiment enables us to fit reliable scaling laws that predict performance across compute budgets from 10^{19} to 10^{22} FLOPs.

Experimental grid. We systematically vary two key dimensions, using 53 model configurations spanning:

- **Model sizes:** 50M, 91M, 175M, 343M, 790M, 1.6B, 3.1B, 6.6B parameters
- **Training tokens:** 1.1B, 2.3B, 4.7B, 9.4B, 19B, 38B, 76B, 152B tokens

This grid provides comprehensive coverage of the (model size, training tokens) parameter space to allow fitting the Chinchilla-style scaling laws described in Equation 2.

Training configuration. All models use an architecture similar to the one described in Section B.3, but without grouped-query attention (GQA) for simplicity. Model width and depth vary to match the desired parameter count, and the learning rate is set to 10^{-2} for all models as we use μ P-simple. After training, we measure the pretraining dataset validation loss, benchmark accuracy, and bits-per-byte (BPB) on the golden answers for each benchmark. These measurements form the basis of our two-stage prediction pipeline: first predicting BPB from model size and training tokens, then mapping BPB to task accuracy.

MMLU reformulation for stable scaling. In standard MMLU evaluation [Hendrycks et al., 2020], models are shown multiple-choice questions and must answer by reference (i.e., select from options labeled A, B, C, or D). This format produces inconsistent and unpredictable performance patterns across different model sizes [Schaeffer et al., 2024], making it difficult to forecast how larger models will perform. Following Bhagia et al. [2024], we reformulate MMLU by removing the answer-by-reference requirement. Instead, we present only the question and evaluate the model’s log-likelihood of each option text directly. We call this variant “MMLU-Direct”.

Figure 11 shows the impact of this reformulation. The original (i.e. by reference) MMLU format yields irregular accuracy-to-BPB relationships (left), while MMLU-Direct shows smooth, predictable scaling (center). The two formats remain strongly correlated (right), confirming that MMLU-Direct preserves the benchmark’s discriminative power while enabling cleaner scaling analysis. However, MMLU-Direct typically produces lower absolute accuracy scores for models above 10^{21} FLOPs. All subsequent references to MMLU in our scaling analysis use this reformulated version.

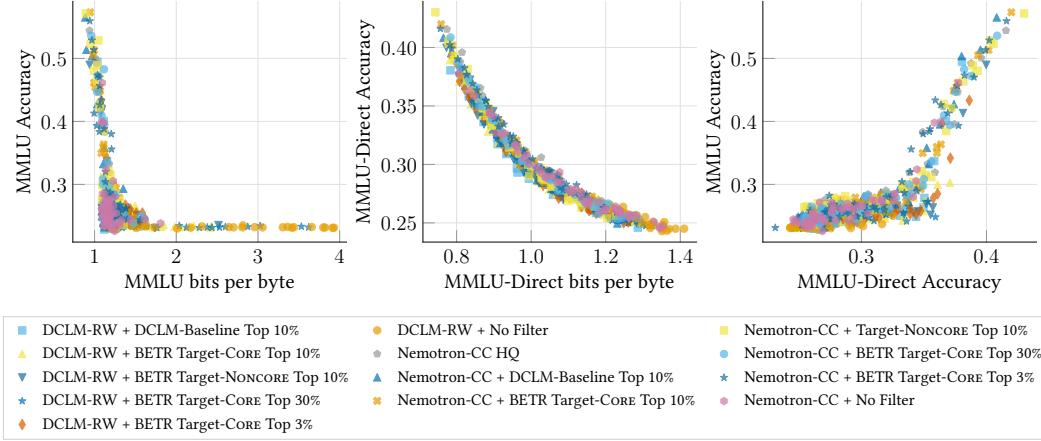


Figure 11: **MMLU task accuracy vs. bits-per-byte scaling.** Comparison between the original MMLU evaluation format (i.e. by reference) and the reformulated MMLU-Direct variant. **Left:** Original MMLU accuracy as a function of bits-per-byte shows irregular patterns. **Middle:** MMLU-Direct accuracy as a function of bits-per-byte shows a clean, predictable relationship. **Right:** Direct comparison of accuracy between the original and reformulated versions, showing a strong monotonic relationship. We use MMLU-Direct in all scaling law analyses for its cleaner scaling properties.

C.2 Batch size selection

Large batch sizes hurt model performance beyond a critical threshold [McCandlish et al., 2018]. For fair scaling comparisons, we must ensure all models train below this threshold. We follow the critical batch size scaling law from Zhang et al. [2024a], which found that the critical token batch size B^* depends primarily on total training tokens D :

$$B^* = d \cdot D^{-\gamma}, \quad (1)$$

where $\gamma = -0.47$ and $d = 22.91$ based on their empirical measurements. To remain safely below the critical threshold, we apply a 20% reduction from B^* , then round down to the nearest power of 2 for implementation convenience. Figure 12 illustrates this selection procedure across our range of training tokens (1B to 152B).

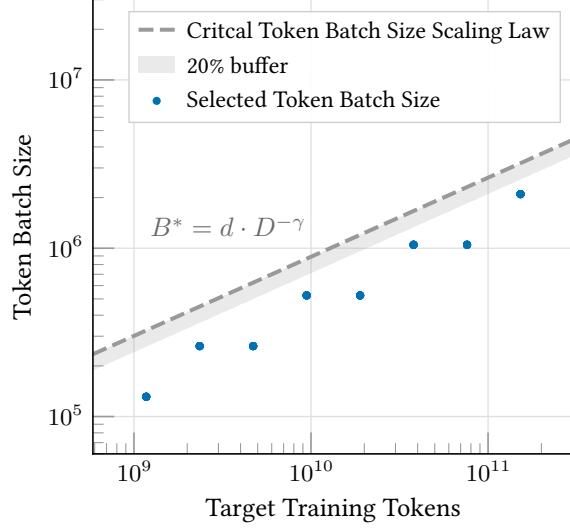


Figure 12: **Batch size selection for scaling experiments.** The dashed gray line shows the critical batch size B^* as a function of training tokens following [Zhang et al. \[2024a\]](#). The light gray region indicates our 20% safety buffer below B^* . Blue circles mark our selected batch sizes, which are rounded down to the nearest power of 2 from the buffer threshold. This ensures all models train below their critical batch size.

C.3 Loss scaling with model size and training tokens

With all models trained and evaluated, we now fit scaling laws to predict loss at arbitrary compute budgets. This forms the first stage of our two-stage prediction pipeline.

We fit scaling laws of the form:

$$L_i(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}, \quad (2)$$

where L_i is either validation loss or benchmark-specific BPB, N is model size (total parameters), D is training tokens, and A, B, E, α, β are fitted parameters. This functional form, introduced by [Hoffmann et al. \[2022\]](#), captures how loss improves with both model size and training tokens.

Following [Besiroglu et al. \[2024\]](#), we fit these parameters by minimizing the Huber loss between predicted and observed values:

$$\min_{a,b,e,\alpha,\beta} \sum_{\text{Run } i} \text{Huber}_\delta (\text{LSE}(a - \alpha \log N_i, b - \beta \log D_i, e) - \log L_i), \quad (3)$$

where LSE denotes log-sum-exp and $\delta = 10^{-3}$. To improve numerical stability and ensure positivity of parameters, we reparameterize the optimization in log space as $a = \log A$, $b = \log B$, and $e = \log E$, and directly optimize over (a, b, α, β, e) . We use the BFGS optimizer implemented in `scipy.optimize.minimize`, beginning with a grid search over initial parameter values to identify good starting points. Finally, we perform 4000 bootstrap resamplings of the training data points (with replacement) to generate a distribution over fitted curves, enabling uncertainty estimation in downstream predictions.

Figure 13 shows the fitted loss surface (Equation 2) for one dataset. To identify compute-optimal models from this fit, we use the standard approximation $C = 6ND$ (FLOPs = 6 × parameters × tokens) to find $N_{opt}(C)$ (i.e. the model size that minimizes loss for each compute budget [[Hoffmann](#)

et al., 2022]). By binning the FLOPs range with logarithmic steps, we identify which trained models lie closest to this compute-optimal curve (Figure 13, left).

Since our grid sampling creates higher density in mid-range FLOPs, we apply inverse density-based reweighting by bins in log-FLOPs space to ensure unbiased fits across all scales. Figure 14 shows example fits for each benchmark, revealing that each has slightly different compute-optimal configurations. We therefore use validation loss to define compute-optimality for each dataset, as it provides a consistent reference across benchmarks.

We repeat this fitting procedure for validation loss and each benchmark’s BPB across all datasets. Figure 15 compares these fits across different data selection methods. Table 12 provides selected fitted parameters.

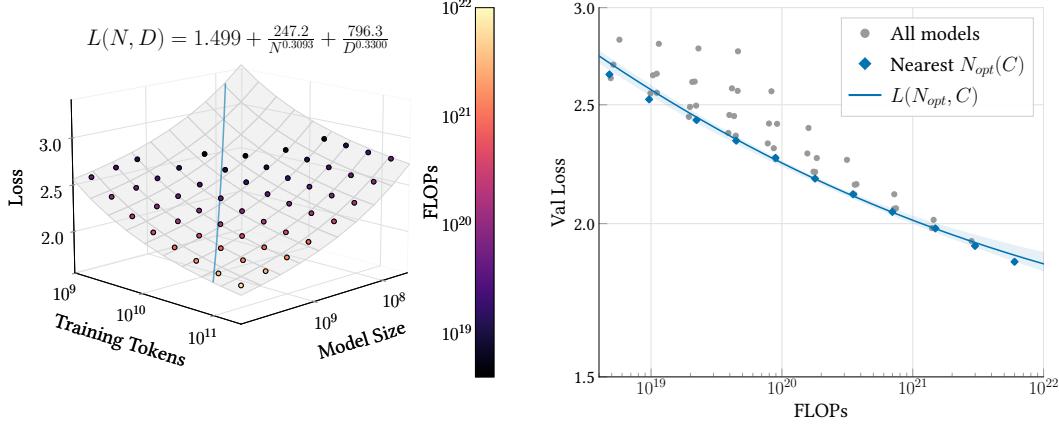


Figure 13: Validation loss scaling with model size and training tokens. Left: Validation loss surface as a function of model size and training FLOPs for models trained on BETR Target-CORE (Nemotron-CC, top 10% filtering). Contours show fitted scaling law values; the blue curve shows the compute-optimal model size $N_{opt}(C)$. **Right:** Validation loss along the compute-optimal path: gray dots show all trained models, blue highlights those closest to compute-optimal, and the line shows our fitted scaling curve $L(N_{opt}, C)$.

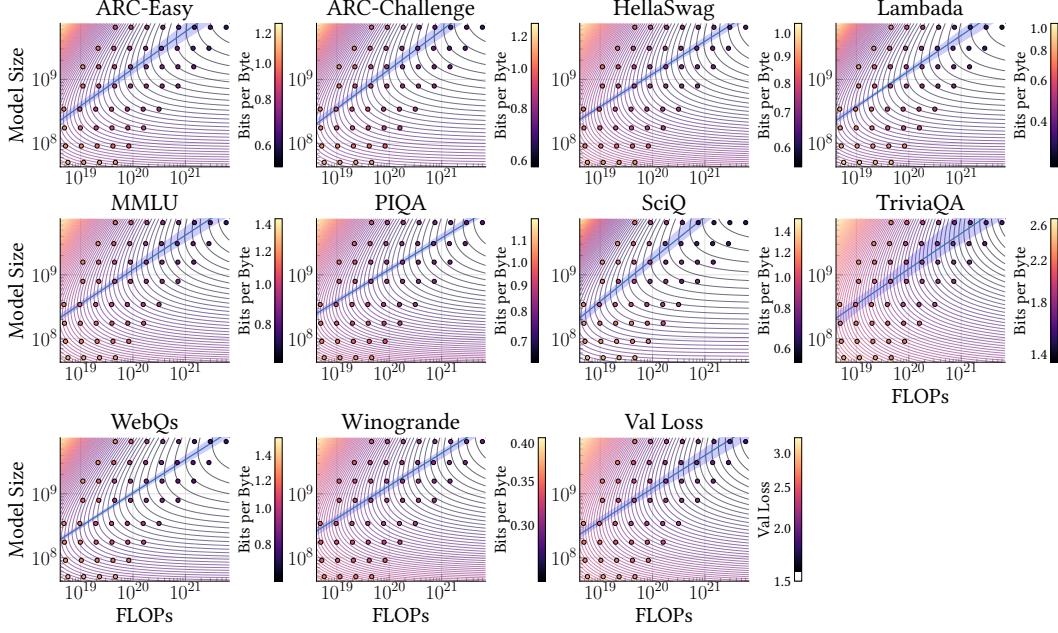


Figure 14: Task loss scaling with model size and training tokens. Bits-per-byte loss surfaces for individual benchmarks and pretraining validation loss (bottom right) for BETR Target-CORE (Nemotron-CC, top 10% filtering). Each panel shows how we fit scaling laws to every benchmark individually, with blue curves showing compute-optimal trajectories $N_{\text{opt}}(C)$. These benchmark-specific fits enable the accuracy predictions shown in the main paper.

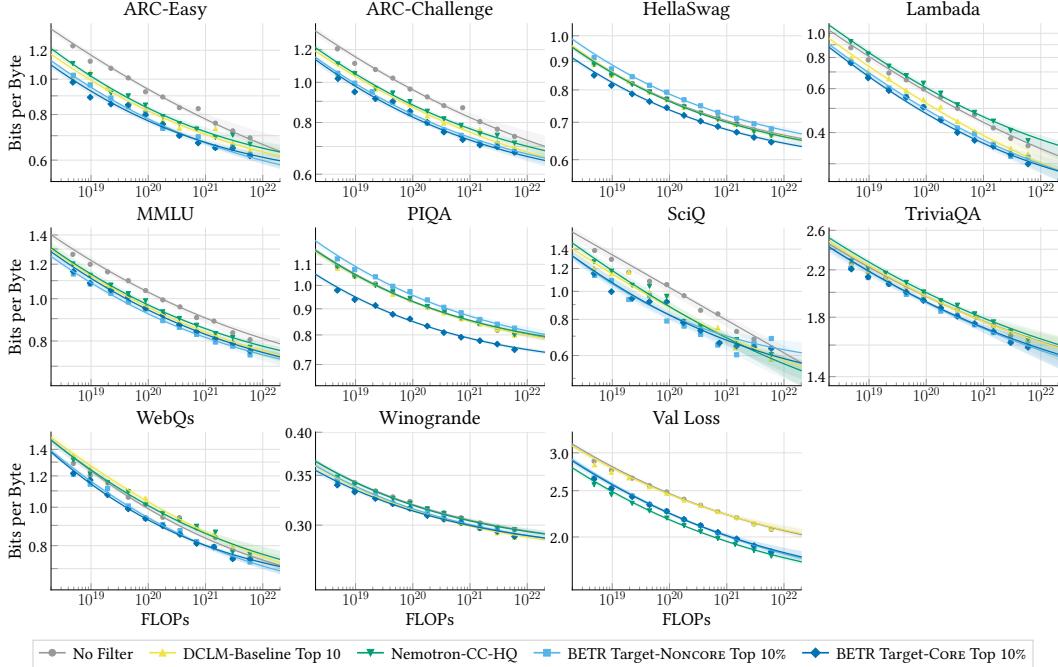


Figure 15: Task loss scaling comparison across data selection methods. Compute-optimal performance on each benchmark for models trained on Nemotron-CC with different data selection methods. Lines show fitted scaling laws with 95% confidence intervals (shaded). These per-benchmark fits show how each data selection method scales differently across tasks.

C.4 Task accuracy scaling

We follow Llama 3’s methodology [Meta AI, 2024] and model task accuracy by fitting each benchmark’s accuracy as a shifted and scaled sigmoid function of its BPB:

$$\text{Acc}_i(L_i) = \frac{c_1}{1 + e^{-k \cdot (L_i - L_0)}} + c_2. \quad (4)$$

Here Acc_i is the accuracy of a model on benchmark i , L_i is the BPB on the same benchmark. The constants c_1 , c_2 , k , and L_0 are fitting parameters.

For each benchmark task, we fit a separate sigmoid curve to the (L_i, Acc_i) pairs, minimizing the L2 loss between predicted and observed accuracy values. Unlike prior work that uses smoothing of consecutive intermediate checkpoints [Zhang et al., 2024a], we only use accuracy and loss values for the final checkpoints of trained models. Following Bhagia et al. [2024], we append a data point at $L = 0.0$ and $\text{Acc} = 1.0$ to each fit. We apply bootstrapping to estimate uncertainty, generating multiple fits by resampling the dataset (with replacement). This bootstrap distribution is propagated into our two-step prediction pipeline to estimate uncertainty in downstream predictions of task accuracy as a function of model size and training tokens.

Importantly, we perform these fits independently per benchmark and per dataset, as we observe significant variations in the loss-to-accuracy relationship across different data pools, data selection methods, and filtering rates. Figure 16 shows example accuracy-to-loss fits for the Nemotron-CC BETR Target-CORE dataset, while Figure 17 illustrates how these relationships vary across different filtering strategies and intensities. Maintaining independent fits for each configuration is critical for accurate predictions.

Following Bhagia et al. [2024], we combine our loss $L_i(N, D)$ and accuracy $\text{Acc}_i(L_i)$ fits into two-step scaling law predictions of benchmark accuracy directly from model size and training tokens, as visualized in Figure 9.

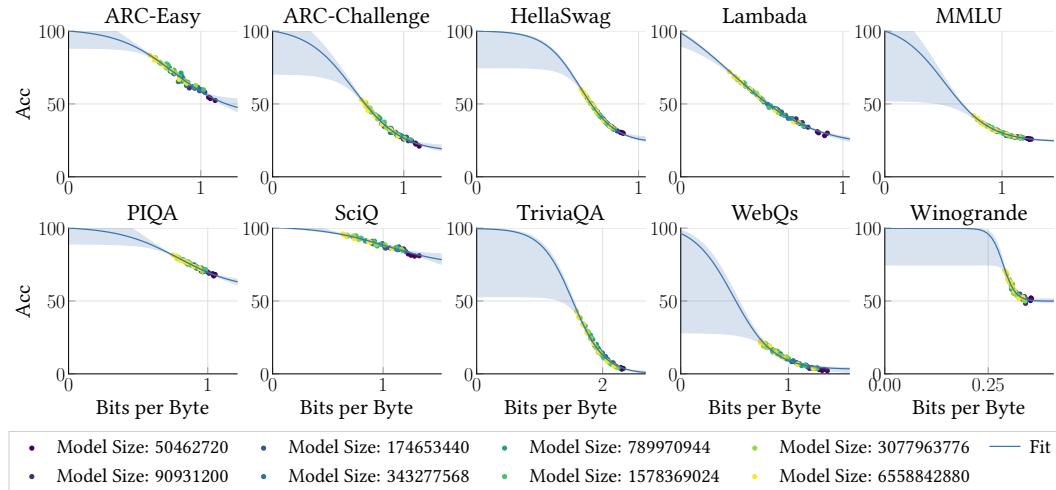


Figure 16: Task accuracy as a function of bits-per-byte. Scaling law fits from the second stage of our two-step prediction pipeline, relating benchmark accuracy to task loss (in bits-per-byte) for models trained on the Nemotron-CC BETR Target-CORE dataset. Each subplot shows data points colored by model size and the corresponding sigmoid fit with 95% bootstrap confidence intervals (blue band). These fits capture the saturating behavior of task accuracy as loss improves and are critical for translating loss scaling predictions into accuracy estimates.

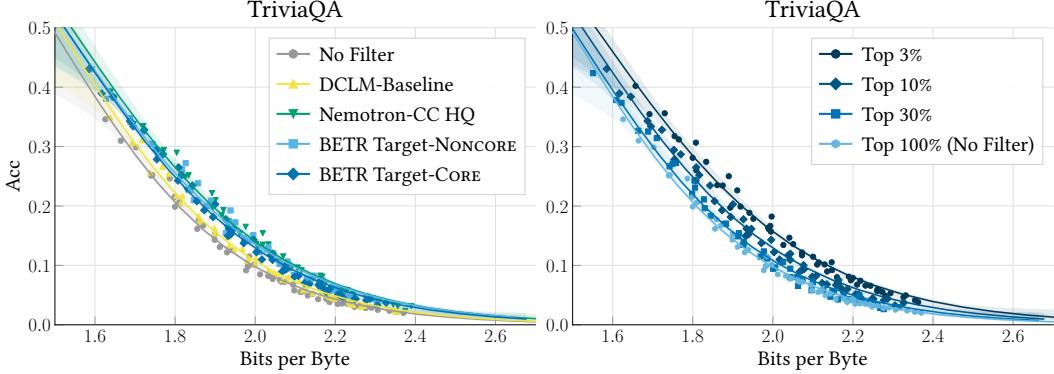


Figure 17: Impact of data selection methods and filtering intensity on accuracy-to-loss scaling. Each plot shows the fitted relationship between TriviaQA bits-per-byte (BPB) and TriviaQA accuracy for models trained on Nemotron-CC with varying selection approaches. **Left:** Comparison across data selection methods. **Right:** Comparison across filtering intensities (Top 3% to 100%) for BETR Target-CORE. Bands denote 95% bootstrap confidence intervals for the sigmoid fits.

C.5 CORE average accuracy computation

To compute the CORE average accuracy reported throughout the paper, we simply average the individual benchmark accuracies:

$$\text{Acc}_{\text{mean}}(N, D) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \text{Acc}_i(L_i(N, D)), \quad (5)$$

where \mathcal{B} denotes the set of CORE benchmarks.

The bootstrap distributions from our two-step prediction pipeline naturally propagate through this averaging, providing confidence intervals for the mean accuracy at any compute budget. This enables us to determine which dataset outperforms another with quantified uncertainty (Figure 1) and to identify optimal filtering rates across scales (Figures 10 and 22).

C.6 Compute multipliers

Throughout the paper, we use compute multipliers (CMs) to compare data selection methods. A compute multiplier is the ratio of baseline compute to method compute needed for the same performance [Betker, 2023; Amodei, 2025]. Visually, this corresponds to a horizontal shift of the scaling curve (e.g., a 2x multiplier means the method’s curve is shifted left by a factor of 2 on the compute axis).

To obtain robust estimates, we average this ratio across the full accuracy range. We:

1. Use fitted scaling laws to predict compute-optimal accuracy vs FLOPs for both methods
2. Discretize predictions into accuracy bins
3. Calculate mean FLOPs required per bin for each method
4. For overlapping bins, compute ratios (baseline FLOPs / method FLOPs)
5. Average these ratios

Table 5 reports the resulting compute multipliers, where values greater than 1 indicate the method outperforms the baseline (here, no filtering).

Method	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
Data pool: DCLM-RefinWeb											
No Filter	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DCLM-Baseline	10.2	8.7	1.2	1.9	5.7	1.0	3.2	1.9	1.1	1.7	2.6
BETR Target-NONCORE	13.5	10.2	0.7	2.8	7.9	0.5	5.7	3.4	1.4	1.5	3.1
BETR Target-CORE	17.9	13.8	3.1	3.6	5.7	9.7	5.3	3.0	1.3	1.9	4.7
Data pool: Nemotron-CC											
No Filter	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Nemotron-CC HQ	3.7	3.5	1.3	0.7	3.8	1.2	2.7	2.0	1.0	1.3	1.7
DCLM-Baseline	3.2	3.1	1.1	2.2	3.3	0.9	2.9	1.3	1.0	1.9	1.9
BETR Target-NONCORE	6.1	5.2	0.6	3.3	6.6	0.4	5.0	4.4	3.2	1.5	3.0
BETR Target-CORE	10.4	8.2	2.7	4.5	5.2	9.5	6.9	3.6	2.8	1.8	4.7

Table 5: **Compute multipliers for BETR and baselines.** Compute multipliers relative to no filtering baseline for each data pool. BETR Target-Core directly optimizes for the evaluation benchmarks and achieves the highest average compute multiplier. **Bold:** Highest compute multiplier per benchmark for each data pool.

D Additional scaling law findings

D.1 Compute-optimal token-to-parameter ratios

Figure 18 shows the optimal training tokens to model size ratio D/N_{opt} as a function of compute across different data pools and selection methods. While Hoffmann et al. [2022] identified an optimal ratio of ~ 20 tokens per parameter, our models consistently exhibit lower values, typically around 10-12 tokens per parameter.

This ratio remains stable: we observe no significant trends with increasing compute, and different data selection methods yield similar ratios within uncertainty bounds. This indicates that a constant token-to-parameter ratio may suffice across the compute scales we study, regardless of the data selection method used.

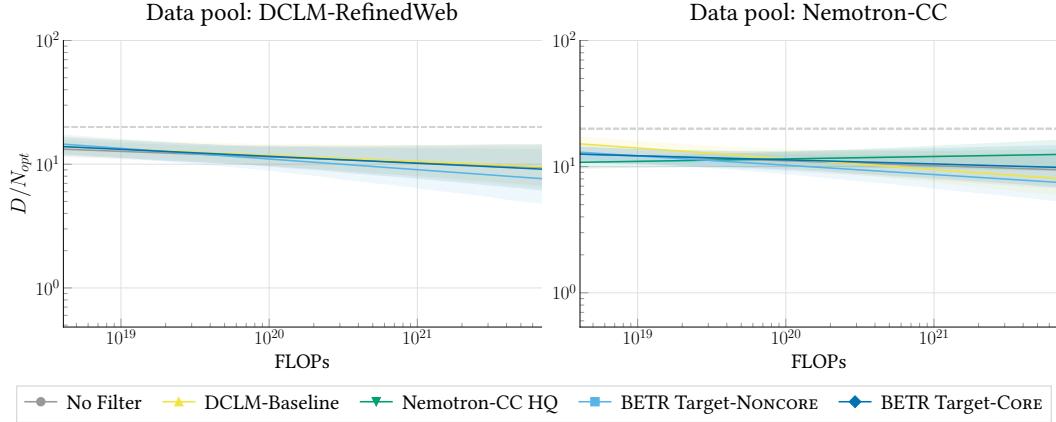


Figure 18: **Optimal training token to model size ratio D/N_{opt} across compute scales.** Ratio of training tokens D to compute-optimal model size $N_{\text{opt}}(C)$ as a function of compute (FLOPs) for models trained on DCLM-RefinWeb (left) and Nemotron-CC (right) data pools. Lines show the ratio for each data selection method, with shaded regions denoting 95% confidence intervals from bootstrap resampling. The dashed horizontal line marks the Chinchilla ratio of $D/N \approx 20$ [Hoffmann et al., 2022]. All methods yield ratios around 10-12 tokens per parameter.

D.2 Per-benchmark scaling laws

Figures 19 and 20 show per-benchmark accuracy scaling for compute-optimal models on DCLM-RefinedWeb and Nemotron-CC data pools, respectively. Across nearly all benchmarks, BETR Target-CORE consistently achieves the highest performance.

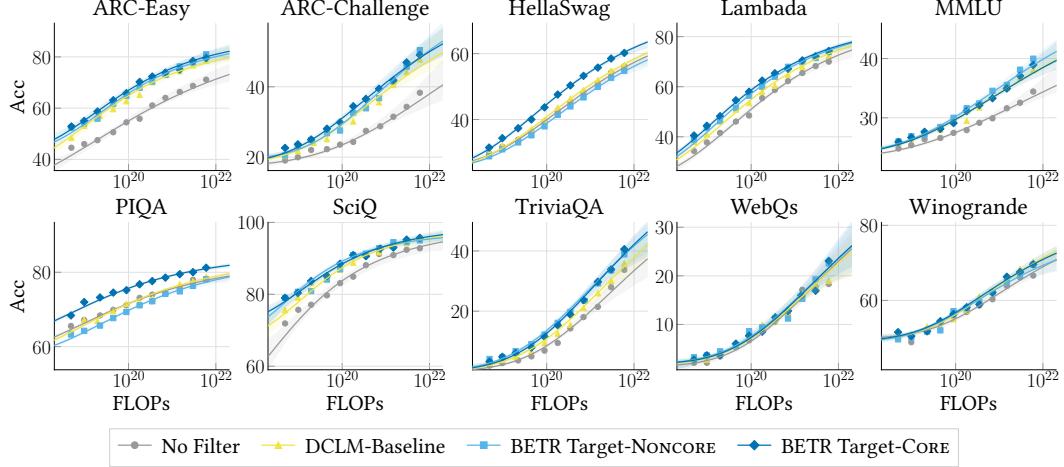


Figure 19: **Per-benchmark accuracy scaling on DCLM-RefinedWeb.** Compute-optimal task accuracy for models trained on DCLM-RefinedWeb with different data selection methods.

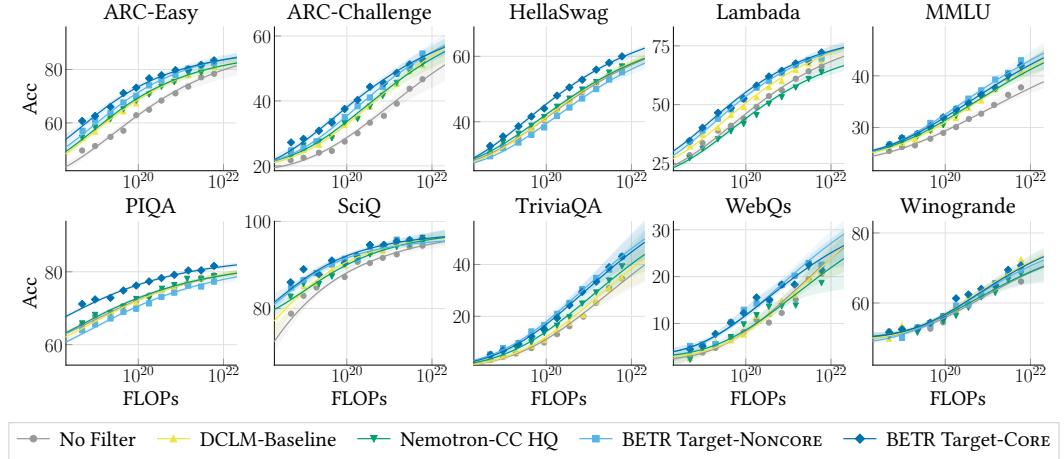


Figure 20: **Per-benchmark accuracy scaling on Nemotron-CC.** Compute-optimal task accuracy for models trained on Nemotron-CC with different data selection methods.

D.3 Data pool comparison

Figure 21 directly compares CORE average accuracy across both data pools. Without filtering, Nemotron-CC consistently outperforms DCLM-RefinedWeb at all compute budgets. BETR Target-CORE filtering improves both data pools, with Nemotron-CC + BETR achieving the best overall performance. The scaling laws suggest the gap between data pools may narrow at higher compute budgets.

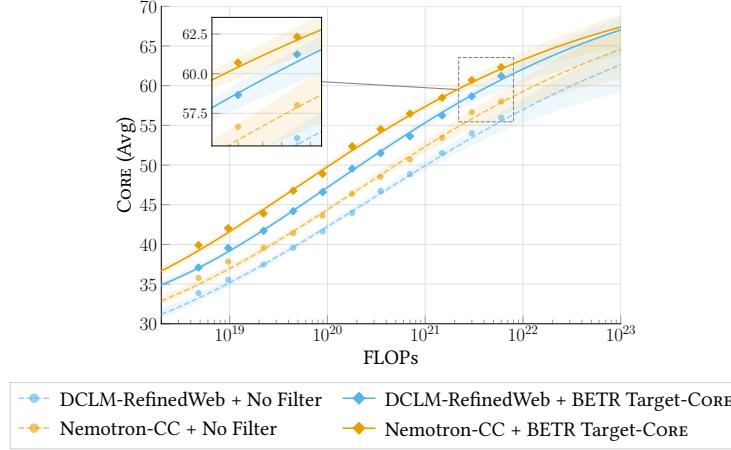


Figure 21: **Data pool comparison with and without BETR filtering.** Mean benchmark accuracy (CORE average) as a function of compute for models trained on DCLM-RefinedWeb (orange) and Nemotron-CC (blue), with (solid) and without (dashed) BETR Target-CORE filtering. Lines show scaling law predictions with 95% confidence intervals; markers indicate actual trained models closest to compute-optimal.

D.4 Optimal filtering rates on DCLM-RefinedWeb

Figure 22 extends our filtering rate analysis to the DCLM-RefinedWeb data pool. As with Nemotron-CC (Figure 10), we observe a shift from aggressive filtering (Top 3%) toward lighter filtering (Top 10%) as compute increases. However, the transition is less pronounced: the optimal filtering rate scales as $F_{\text{opt}} \propto C^{0.14}$, compared to $C^{0.25}$ for Nemotron-CC. At 10^{23} FLOPs, the optimal filtering rate is predicted to be Top 10%, compared to Top 30% for Nemotron-CC at the same compute budget.

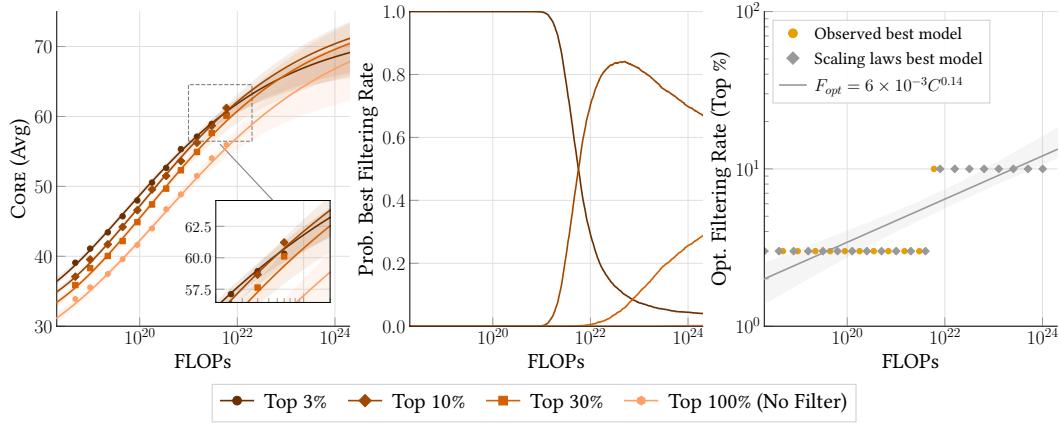


Figure 22: **Optimal filtering rate scaling on DCLM-RefinedWeb.** **Left:** Mean task accuracy (CORE average) for models trained with different BETR Target-CORE filtering rates, with scaling law fits and 95% confidence intervals. **Middle:** Probability that each filtering rate is optimal at a given compute scale, estimated from bootstrap distributions. **Right:** Optimal filtering rate (top-x%) as a function of compute for observed models (orange) and scaling law predictions (gray). The fitted relationship $F_{\text{opt}}(C) = 6 \times 10^{-3}C^{0.14}$ is shown.

D.5 Fixed model size scaling

Figure 23 compares mean benchmark accuracy across data selection methods for a fixed model size of 6.6B parameters as a function of training tokens. This analysis extends to the overtraining regime commonly used in production LLMs. The results show that the performance ranking observed under compute-optimal scaling persists at fixed model size: BETR Target-CORE consistently achieves the highest performance, outperforming DCLM-Baseline, Nemotron-CC HQ, and unfiltered data across all training budgets.

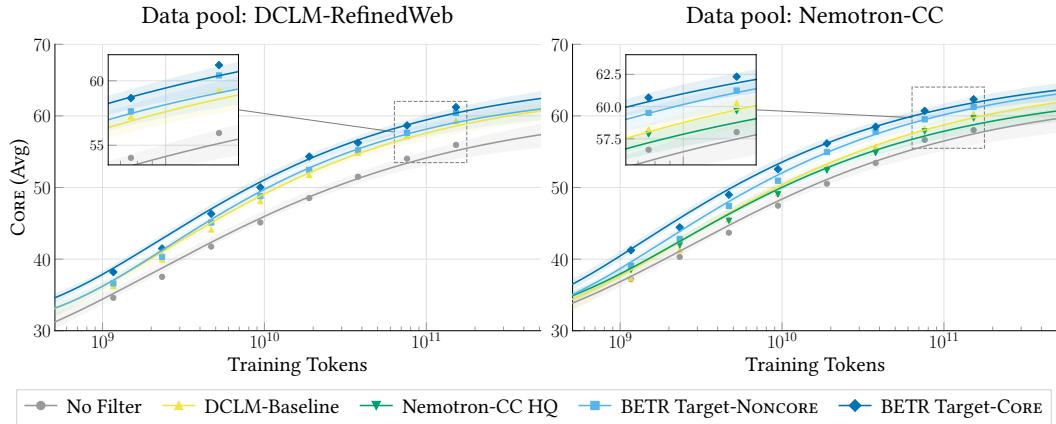


Figure 23: **Performance comparison at fixed model size.** Mean benchmark accuracy (CORE average) as a function of training tokens for 6.6B parameter models trained on DCLM-RefinedWeb (left) and Nemotron-CC (right) with various data selection methods. Lines show scaling law fits; markers indicate measured performance. BETR Target-CORE consistently outperforms baselines across all token counts. Inset plots magnify the high-token regime to highlight performance differences.

Figure 24 investigates how optimal filtering rates change with training tokens at fixed 6.6B model size. Unlike compute-optimal scaling, we do not observe a transition from 10% to 30% filtering within the sub- 10^{24} FLOPs range. However, a transition from 3% to 10% filtering still occurs as training progresses. The optimal filtering rate scales as $F_{\text{opt}}(D) \propto D^{0.13}$, slower than the $F_{\text{opt}}(C) \propto C^{0.25}$ observed under compute-optimal conditions. This shows that optimal filtering strategies differ between fixed model size scaling and compute-optimal scaling.

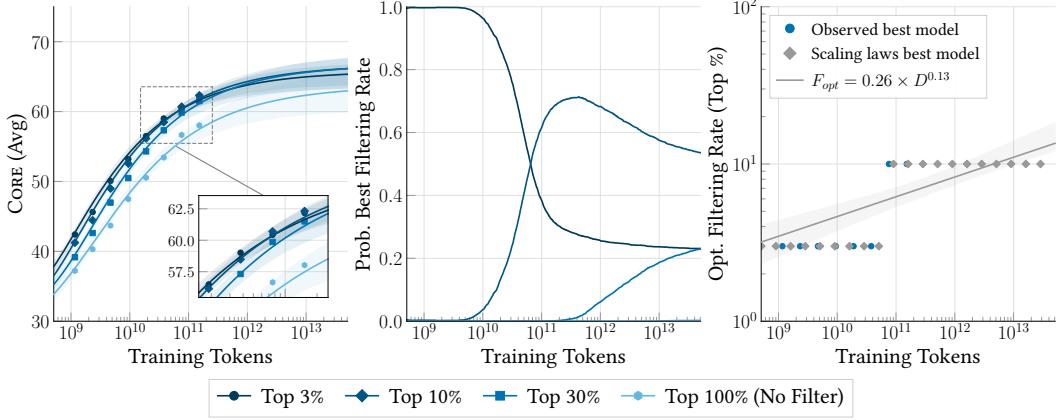


Figure 24: Optimal data filtering rate scaling at fixed model size. **Left:** Mean task accuracy (CORE average) for 6.6B models trained with different BETR Target-CORE filtering rates on Nemotron-CC, along with scaling law fits. **Center:** Probability that each filtering rate is optimal at a given number of training tokens, estimated from bootstrap distributions of scaling law fits. **Right:** Optimal filtering rate (top- $x\%$) as a function of training tokens, shown both for the **best observed model** and the best model predicted via scaling laws. The scaling law fit $F_{opt}(D) = 0.26 \times D^{0.13}$ is overlaid for comparison.

D.6 Example scaling law parameters

Tables 12 and 13 present example scaling law parameters for models trained on Nemotron-CC with BETR Target-CORE filtering at various filtering rates. For brevity, we show only a representative subset of the complete parameter fits. Each parameter includes 95% confidence intervals derived from bootstrap distributions. Note that these parameter-level uncertainties do not directly translate to the prediction uncertainties shown in our plots, which are computed from the full ensemble of bootstrap models.

E Additional experiments & analysis

E.1 Detailed NONCORE results

To complement Figure 6, we report the full NONCORE results at 7B-10x scale in Table 6. The benchmarks in each subcategory are listed in Table 11.

Method	NONCORE (Avg)	NONCORE Subcategories				
		Commonsense Reasoning	Language Understanding	Reading Comprehension	Symbolic Problem Solving	World Knowledge
Data pool: DCLM-RefinedWeb						
No Filter	44.2	69.1	44.2	62.5	22.5	34.7
DCLM-Baseline	46.3	71.4	49.8	65.2	23.4	36.6
BETR Target-NONCORE	46.4	71.2	47.7	65.9	23.6	37.6
BETR Target-CORE	44.2	69.8	47.7	62.8	20.8	35.8
Data pool: Nemotron-CC						
No Filter	45.0	70.8	47.2	62.2	22.7	37.3
Nemotron-CC HQ	47.7	72.8	50.4	68.2	23.7	39.9
DCLM-Baseline	50.4	73.4	53.6	70.0	28.7	40.0
BETR Target-NONCORE	50.6	74.1	54.1	70.6	28.1	40.3
BETR Target-CORE	48.5	73.9	50.9	67.9	25.5	38.5

Table 6: **NONCORE results at 7B-10x scale.** We compare BETR variants and baselines at 7B-10x scale (7B parameters, 1.4T tokens) on NONCORE tasks. BETR Target-NONCORE directly optimizes for these benchmarks and achieves the highest average performance, followed closely by DCLM-Baseline. **Bold:** best result.

E.2 7B-1x CORE results

For completeness, we also report CORE performance at 7B-1x scale for BETR variants and baselines in Table 7.

Method	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
Data pool: DCLM-RefinedWeb											
No Filter	72.3	39.0	56.1	71.2	27.4	78.2	94.0	33.7	19.8	68.4	56.0
DCLM-Baseline	79.4	48.1	57.0	72.7	52.8	77.7	95.2	35.7	20.1	71.4	61.0
BETR Target-NONCORE	79.2	48.0	55.0	73.0	53.3	77.0	94.7	38.3	22.2	68.4	60.9
BETR Target-CORE	79.7	49.7	59.9	74.0	48.9	81.3	95.4	40.4	22.5	71.0	62.3
Data pool: Nemotron-CC											
No Filter	78.8	44.9	56.3	65.9	50.7	78.2	95.1	36.3	21.1	67.9	59.5
Nemotron-CC HQ	81.0	51.6	56.8	63.6	58.0	79.4	95.2	36.5	17.5	68.2	60.8
DCLM-Baseline	79.9	51.9	57.2	69.6	56.7	78.9	96.5	36.1	22.2	72.7	62.2
BETR Target-NONCORE	83.0	54.8	55.2	70.0	59.9	77.8	96.2	43.1	22.1	69.0	63.1
BETR Target-CORE	83.4	54.1	59.9	71.5	59.4	81.6	95.8	43.0	25.3	72.2	64.6

Table 7: **CORE results at 7B-1x scale.** Comparison of BETR variants and baselines at 7B-1x scale (7B parameters, 140B tokens). BETR Target-CORE directly optimizes for these benchmarks and achieves the highest average performance. **Bold:** best result ± 1 std.

E.3 Nemotron-CC HQ dataset definition

Throughout this work, we use Nemotron-CC HQ to refer to all documents labeled as “high-quality” from the Nemotron-CC data pool, which includes both real documents and all types of synthetic rewrites. This slightly differs from the dataset used by [Su et al. \[2024a\]](#), which selects only high-quality real documents and high-quality synthetic documents specifically from the diverse QA pairs rewriting style, excluding other synthetic rewriting styles.

To understand the impact of this difference, we compare these two versions along with other Nemotron-CC subsets at 7B-1x scale in Table 8. While [Su et al. \[2024a\]](#)’s more selective version performs slightly better (~ 1 std better) than our broader definition, this improvement is not enough to change method rankings.

Nemotron-CC Subset	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
All	78.8	44.9	56.3	65.9	50.7	78.2	95.1	36.3	21.1	67.9	59.5
Actual Only	75.3	40.7	56.3	67.2	28.8	77.8	93.9	32.5	16.7	67.1	55.6
Synthetic Only	80.5	49.6	55.9	62.8	55.4	78.8	95.7	38.7	21.5	67.6	60.6
HQ (Actual + Synthetic QA) [†]	81.5	51.0	57.2	64.2	58.6	78.5	95.4	37.3	19.9	69.5	61.3
HQ (All) [‡]	81.0	51.6	56.8	63.6	58.0	79.4	95.2	36.5	17.5	68.2	60.8

Table 8: **Performance of different Nemotron-CC data subsets** at 7B-1x scale (7B parameters, 140B tokens). [†]HQ (Actual + Synthetic Diverse QA) corresponds to the subset used in [Su et al. \[2024a\]](#). [‡]HQ (All) is the definition used throughout this paper as Nemotron-CC HQ.

E.4 Per-benchmark standard deviations

To estimate the natural variation in benchmark performance, we trained 10 models with different random initialization and data shuffling on a representative dataset (BETR Target-CORE on DCLM-RefinedWeb at 7B-1x scale). Table 9 reports the standard deviations across these runs.

	ARC-E	ARC-C	HellaS	Lamb.	MMLU	PIQA	SciQ	TrQA	WebQs	Wino.	CORE (Avg)
Std. Dev.	0.49	0.85	0.18	0.48	1.10	0.44	0.32	0.84	1.33	1.15	0.24

Table 9: **Standard deviations of benchmark performance** across 10 runs with different random initialization and data shuffling, using BETR Target-CORE on DCLM-RefinedWeb at 7B-1x scale (7B parameters, 140B tokens).

E.5 Benchmark contributions to document selection

When using BETR with max aggregation, each document in the top 10% is selected because one benchmark example ranks it highly. Table 10 shows which benchmarks’ examples are responsible for selecting documents when targeting all CORE benchmarks jointly and equally (using the same number of target examples per benchmark). The distribution forms two clusters: six benchmarks (TriviaQA, HellaSwag, MMLU, WebQs, WinoGrande, PIQA) each contribute 10–15% of selected documents, while four benchmarks (Lambada, SciQ, ARC-Easy/Challenge) contribute 5–7%. While not uniform, the distribution remains reasonably balanced with all benchmarks contributing meaningful fractions and no single benchmark dominating. This pattern holds consistently across both DCLM-RefinedWeb and Nemotron-CC data pools.

Benchmark	Contribution (%)	
	DCLM-RW	Nemo-CC
TriviaQA	15.1	14.6
HellaSwag	12.3	12.3
MMLU	12.3	13.2
WebQs	12.1	11.3
WinoGrande	12.0	10.7
PIQA	11.3	11.5
Lambada	6.6	5.8
SciQ	6.4	7.0
ARC-Challenge	6.2	7.0
ARC-Easy	5.8	6.5

Table 10: **Benchmark contribution to BETR document selection.** Percentage of top-10% documents attributed to each benchmark when using BETR Target-CORE with max aggregation. Each document is attributed to the benchmark whose example ranks it highest. All benchmarks are equally targeted (10% of total examples each) but contribute differently to top-10% selection. DCLM-RW = DCLM-RefinedWeb. Nemo-CC = Nemotron-CC.

E.6 Topic and format distributions

To show the content composition of different data selection methods, we compute topic and format distributions using WebOrganizer [Wettig et al., 2025] models. These are computed on a subsample of 1M documents from DCLM-RefinedWeb filtered to the top 10% of tokens. Figure 25 shows these distributions, providing a high-level view of content differences between methods.

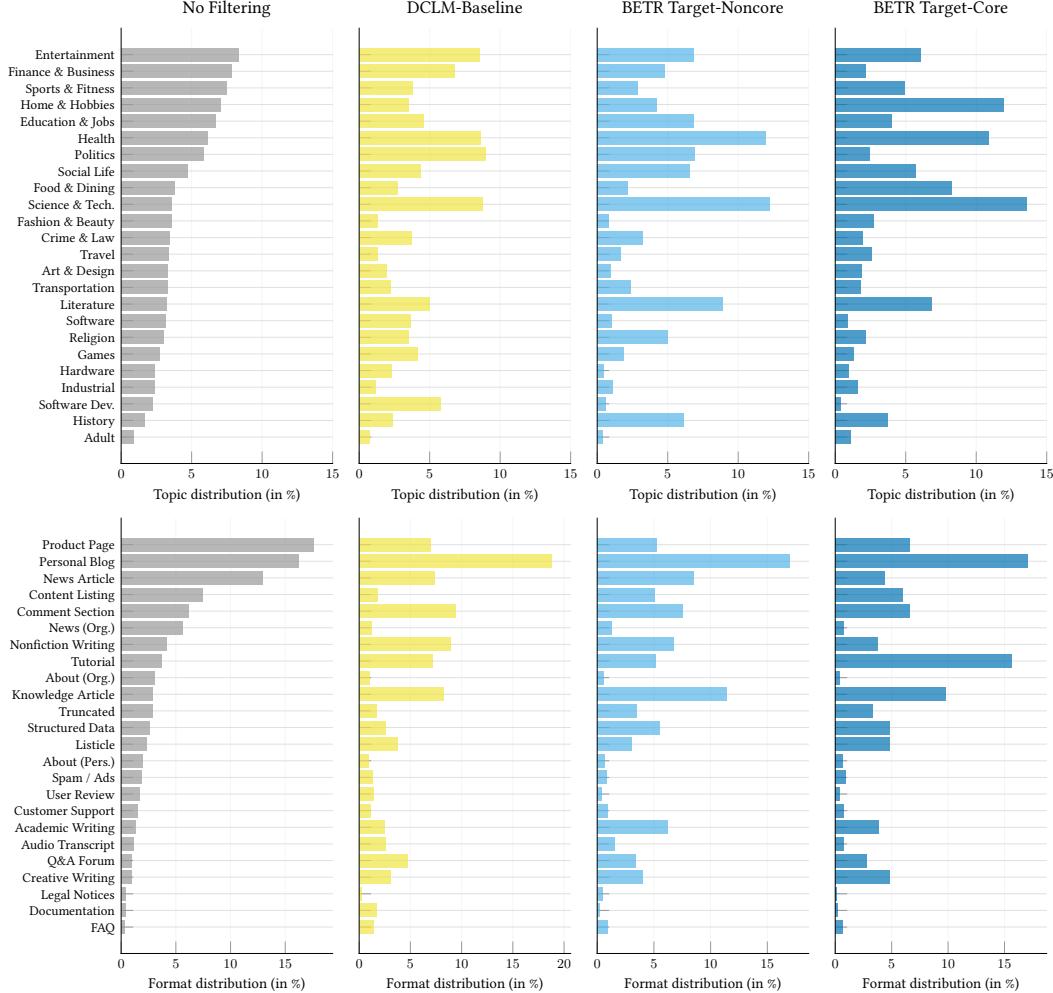


Figure 25: Topic and format distributions on DCLM-RefinedWeb. We show the WebOrganizer [Wettig et al., 2025] topic and format distributions for four datasets: unfiltered DCLM-RefinedWeb, DCLM-Baseline, BETR Target-NONCORE, and BETR Target-CORE (the latter three filtered to top 10% of tokens).

F Full tables

The following tables present the full list of NONCORE benchmarks and example scaling law parameters from our experiments.

Benchmark	Subcategory	Order A	Order B	Source
AGIEval LSAT AR	Symbolic PS	1	39	Zhong et al. [2023]
AGIEval LSAT LR	Reading Comp.	20	5	Zhong et al. [2023]
AGIEval LSAT RC	Reading Comp.	39	2	Zhong et al. [2023]
AGIEval SAT EN	Reading Comp.	39	39	Zhong et al. [2023]
AGIEval SAT Math	Symbolic PS	39	39	Zhong et al. [2023]
AQuA	Symbolic PS	20	10	Ling et al. [2017]
BBQ	Commonsense R.	39	39	Parrish et al. [2021]
BIG-bench Conceptual Combinations	Lang. Und.	20	39	Srivastava et al. [2023]
BIG-bench Conlang Translation	Lang. Und.	20	39	Srivastava et al. [2023]
BIG-bench CS Algorithms	Symbolic PS	39	5	Srivastava et al. [2023]
BIG-bench Dyck Languages	Symbolic PS	20	20	Srivastava et al. [2023]
BIG-bench Elementary Math QA	Symbolic PS	39	5	Srivastava et al. [2023]
BIG-bench Language Identification	Lang. Und.	5	39	Srivastava et al. [2023]
BIG-bench Logical Deduction	Symbolic PS	10	39	Srivastava et al. [2023]
BIG-bench Misconceptions	World Knowledge	10	20	Srivastava et al. [2023]
BIG-bench Novel Concepts	Commonsense R.	39	39	Srivastava et al. [2023]
BIG-bench Operators	Symbolic PS	39	39	Srivastava et al. [2023]
BIG-bench Repeat Copy Logic	Symbolic PS	20	10	Srivastava et al. [2023]
BIG-bench Strange Stories	Commonsense R.	39	20	Srivastava et al. [2023]
BIG-bench Strategy QA	Commonsense R.	39	39	Srivastava et al. [2023]
BIG-bench Understanding Fables	Reading Comp.	10	39	Srivastava et al. [2023]
BoolQ	Reading Comp.	20	20	Clark et al. [2019]
CommonsenseQA	Commonsense R.	20	20	Talmor et al. [2018]
COPA	Commonsense R.	20	10	Roemmele et al. [2011]
CoQA	Reading Comp.	39	39	Reddy et al. [2019]
Enterprise PII Classification	Lang. Und.	39	39	PatronusAI [2023]
GPQA Diamond	World Knowledge	39	20	Rein et al. [2024]
GPQA Main	World Knowledge	5	10	Rein et al. [2024]
GSM8K	Symbolic PS	20	39	Cobbe et al. [2021]
Jeopardy	World Knowledge	10	39	Tunguz [2019]
LogiQA	Symbolic PS	39	39	Liu et al. [2020]
MathQA	Symbolic PS	39	39	Amini et al. [2019]
OpenBookQA	Commonsense R.	39	20	Mihaylov et al. [2018]
PubMedQA	Reading Comp.	39	20	Jin et al. [2019]
Simple Arithmetic No Spaces	Symbolic PS	5	20	MosaicML [2023]
Simple Arithmetic With Spaces	Symbolic PS	39	10	MosaicML [2023]
Social IQA	Commonsense R.	10	39	Sap et al. [2019]
SQuAD	Reading Comp.	2	1	Rajpurkar et al. [2016]
SVAMP	Symbolic PS	39	20	Patel et al. [2021]

Table 11: **List of the 39 NONCORE benchmarks** (from DCLM-Core and DCLM-Extended [Li et al., 2024], excluding overlap with CORE). Order A and Order B indicate each benchmark’s position in two random orderings used to test how performance scales with benchmark diversity (1, 2, 5, 10, 20, and 39 benchmarks) in Figure 5. Commonsense R. = Commonsense Reasoning, Lang. Und. = Language Understanding, Reading Comp. = Reading Comprehension, Symbolic PS = Symbolic Problem Solving.

Dataset	a	b	e	α	β	MAE
Data pool: Nemotron-CC, Filter: No Filter						
ARC-Easy	4.092 (3.370, 5.743)	6.906 (5.607, 9.345)	-0.9782 (-1.248, -0.6162)	0.2454 (0.1992, 0.3478)	0.3628 (0.2961, 0.4821)	0.018
ARC-Challenge	3.853 (2.586, 5.435)	7.026 (4.892, 8.464)	-0.7305 (-1.246, -0.5034)	0.2407 (0.1572, 0.3377)	0.3764 (0.2668, 0.4456)	0.016
HellaSwag	5.001 (4.291, 5.936)	6.544 (6.395, 6.739)	-0.5320 (-0.5638, -0.4977)	0.3519 (0.3092, 0.4071)	0.3864 (0.3794, 0.3965)	0.005
Lambada	5.669 (3.632, 6.944)	7.834 (5.083, 9.892)	-1.787 (-12.92, -1.340)	0.3399 (0.2196, 0.4161)	0.4089 (0.2746, 0.5129)	0.012
MMLU	5.039 (2.832, 6.256)	5.620 (4.617, 8.338)	-0.5209 (-0.9907, -0.3759)	0.3157 (0.1740, 0.3851)	0.3034 (0.2514, 0.4412)	0.015
PIQA	6.092 (5.224, 6.751)	6.314 (6.044, 7.174)	-0.3418 (-0.3790, -0.3068)	0.3999 (0.3489, 0.4389)	0.3624 (0.3477, 0.4064)	0.006
SciQ	3.550 (2.895, 4.413)	5.473 (3.696, 8.820)	-106.1 (-106.1, -106.1)	0.1940 (0.1538, 0.2474)	0.2707 (0.1831, 0.4357)	0.043
TriviaQA	3.427 (1.969, 5.385)	6.520 (4.686, 7.509)	0.1640 (-2.863, 0.3539)	0.1962 (0.07290, 0.3200)	0.3339 (0.2390, 0.3828)	0.023
WebQs	5.922 (4.587, 7.566)	8.420 (5.951, 9.234)	-0.5823 (-0.7948, -0.4336)	0.3526 (0.2759, 0.4494)	0.4299 (0.3079, 0.4704)	0.018
WinoGrande	3.449 (3.448, 3.449)	4.840 (4.839, 4.840)	-1.291 (-1.300, -1.278)	0.3424 (0.3384, 0.3478)	0.3738 (0.3713, 0.3782)	0.002
Val Loss	5.450 (4.608, 6.971)	6.676 (6.174, 7.641)	0.5218 (0.4453, 0.6148)	0.3037 (0.2518, 0.3945)	0.3282 (0.3029, 0.3780)	0.022
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 3%						
ARC-Easy	4.480 (3.464, 5.976)	8.105 (5.881, 8.658)	-0.8082 (-1.055, -0.6930)	0.2954 (0.2310, 0.3834)	0.4472 (0.3341, 0.4739)	0.012
ARC-Challenge	4.651 (3.681, 5.474)	7.431 (6.411, 7.951)	-0.6484 (-0.7877, -0.5992)	0.3095 (0.2503, 0.3578)	0.4150 (0.3631, 0.4409)	0.009
HellaSwag	5.111 (4.374, 5.955)	6.630 (6.432, 6.764)	-0.5466 (-0.5779, -0.5238)	0.3614 (0.3172, 0.4105)	0.3943 (0.3842, 0.4013)	0.004
Lambada	6.923 (5.576, 7.801)	10.45 (8.896, 11.97)	-1.650 (-1.832, -1.482)	0.4176 (0.3431, 0.4697)	0.5401 (0.4650, 0.6120)	0.011
MMLU	4.359 (3.525, 6.152)	6.243 (5.439, 8.014)	-0.5874 (-0.7090, -0.4056)	0.2830 (0.2308, 0.3919)	0.3424 (0.3021, 0.4326)	0.012
PIQA	5.565 (5.565, 6.239)	6.913 (6.781, 6.914)	-0.3924 (-0.4109, -0.3776)	0.3858 (0.3821, 0.4247)	0.4083 (0.4018, 0.4102)	0.006
SciQ	6.735 (3.341, 10.20)	15.48 (9.509, 19.85)	-0.6378 (-1.523, -0.5290)	0.3992 (0.1906, 0.5964)	0.7822 (0.4912, 0.9972)	0.045
TriviaQA	3.732 (2.215, 5.297)	7.449 (5.828, 8.053)	0.2570 (-0.4100, 0.3792)	0.2174 (0.1025, 0.3161)	0.3809 (0.2996, 0.4103)	0.021
WebQs	7.586 (5.926, 8.412)	8.494 (7.333, 9.468)	-0.4430 (-0.5773, -0.4059)	0.4591 (0.3637, 0.5053)	0.4383 (0.3803, 0.4862)	0.013
WinoGrande	3.301 (3.301, 3.302)	4.951 (4.951, 4.951)	-1.296 (-1.306, -1.284)	0.3379 (0.3315, 0.3420)	0.3810 (0.3774, 0.3849)	0.002
Val Loss	5.291 (4.456, 7.190)	6.294 (5.409, 7.744)	0.2539 (0.1408, 0.3988)	0.2980 (0.2462, 0.4099)	0.3127 (0.2662, 0.3873)	0.022
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 10%						
ARC-Easy	5.739 (4.036, 7.209)	9.396 (8.198, 10.96)	-0.6646 (-0.8457, -0.5862)	0.3626 (0.2611, 0.4485)	0.5026 (0.4441, 0.5757)	0.013
ARC-Challenge	4.864 (3.799, 5.869)	9.246 (8.367, 9.957)	-0.6004 (-0.7120, -0.5320)	0.3151 (0.2497, 0.3758)	0.4953 (0.4523, 0.5308)	0.010
HellaSwag	5.168 (4.615, 5.963)	6.428 (6.260, 6.680)	-0.5568 (-0.5757, -0.5278)	0.3656 (0.3337, 0.4128)	0.3838 (0.3758, 0.3979)	0.004
Lambada	6.778 (5.227, 7.403)	9.920 (7.617, 10.92)	-1.620 (-2.073, -1.502)	0.4110 (0.3230, 0.4468)	0.5171 (0.4049, 0.5663)	0.008
MMLU	5.188 (4.390, 6.324)	7.128 (5.988, 8.605)	-0.5057 (-0.6094, -0.4391)	0.3295 (0.2812, 0.3948)	0.3847 (0.3263, 0.4587)	0.011
PIQA	6.086 (4.986, 6.086)	6.907 (6.754, 6.907)	-0.3808 (-0.4032, -0.3677)	0.4104 (0.3488, 0.4149)	0.4019 (0.3942, 0.4065)	0.006
SciQ	5.417 (3.766, 6.138)	12.29 (9.624, 15.88)	-0.8275 (-1.434, -0.7324)	0.3180 (0.2140, 0.3593)	0.6231 (0.4914, 0.7901)	0.030
TriviaQA	3.983 (2.306, 5.237)	5.637 (4.719, 7.557)	0.1363 (-0.6256, 0.3053)	0.2305 (0.1093, 0.3098)	0.2873 (0.2378, 0.3837)	0.023
WebQs	7.187 (6.225, 7.731)	9.279 (7.899, 9.725)	-0.4878 (-0.6013, -0.4695)	0.4332 (0.3758, 0.4633)	0.4754 (0.4060, 0.4961)	0.012
WinoGrande	3.568 (3.567, 3.568)	4.647 (4.647, 4.647)	-1.299 (-1.307, -1.287)	0.3541 (0.3515, 0.3610)	0.3681 (0.3651, 0.3720)	0.002
Val Loss	5.510 (4.608, 7.076)	6.680 (6.067, 7.709)	0.4046 (0.3156, 0.5033)	0.3093 (0.2536, 0.4027)	0.3300 (0.2981, 0.3835)	0.021
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 30%						
ARC-Easy	6.079 (4.420, 6.847)	8.040 (6.119, 9.260)	-0.6383 (-0.8433, -0.5876)	0.3800 (0.2805, 0.4242)	0.4258 (0.3319, 0.4834)	0.013
ARC-Challenge	4.437 (3.762, 5.761)	7.637 (6.121, 8.494)	-0.6142 (-0.7315, -0.5305)	0.2883 (0.2471, 0.3668)	0.4129 (0.3369, 0.4543)	0.011
HellaSwag	5.057 (4.507, 5.779)	6.690 (6.510, 6.808)	-0.5476 (-0.5713, -0.5252)	0.3577 (0.3252, 0.3999)	0.3968 (0.3879, 0.4030)	0.004
Lambada	6.736 (4.803, 7.601)	9.343 (8.454, 11.64)	-1.558 (-2.021, -1.357)	0.4061 (0.2953, 0.4571)	0.4882 (0.4441, 0.6004)	0.012
MMLU	4.501 (4.033, 5.774)	7.398 (6.430, 8.098)	-0.5326 (-0.5753, -0.4419)	0.2874 (0.2582, 0.3623)	0.3959 (0.3494, 0.4307)	0.011
PIQA	5.799 (4.905, 5.800)	6.843 (6.587, 6.957)	-0.3681 (-0.4037, -0.3592)	0.3904 (0.3380, 0.3935)	0.3944 (0.3809, 0.4002)	0.005
SciQ	3.811 (3.170, 6.230)	7.890 (5.179, 11.24)	-2.408 (-14.80, -0.8117)	0.2092 (0.1697, 0.3603)	0.3958 (0.2607, 0.5617)	0.032
TriviaQA	3.257 (2.282, 5.783)	5.930 (4.151, 8.014)	0.007347 (-1.014, 0.3278)	0.1819 (0.1058, 0.3435)	0.3026 (0.2081, 0.4092)	0.025
WebQs	6.101 (4.988, 7.933)	7.988 (6.765, 9.796)	-0.6492 (-0.7833, -0.4710)	0.3666 (0.3012, 0.4707)	0.4063 (0.3482, 0.4963)	0.017
WinoGrande	2.922 (2.922, 2.922)	6.419 (6.419, 6.419)	-1.299 (-1.304, -1.287)	0.3148 (0.3133, 0.3195)	0.4550 (0.4519, 0.4597)	0.001
Val Loss	5.350 (4.554, 6.861)	6.828 (6.298, 7.582)	0.5015 (0.4257, 0.5969)	0.2975 (0.2482, 0.3881)	0.3362 (0.3103, 0.3757)	0.022

Table 12: **Loss scaling law parameters** for BETR Target-CORE and no filtering baseline on the Nemotron-CC data pool. Each column contains the estimated value and the 95% confidence range (lower bound, upper bound). The last column shows the mean average error (MAE) for the fit evaluated on the training date.

Dataset	c_1	c_2	k	L_0	MAE
Data pool: Nemotron-CC, Filter: No Filter					
ARC-Easy	0.6722 (0.4884, 1.0000)	0.3558 (0.2785, 0.4079)	-3.749 (-5.239, -2.582)	0.8376 (0.6798, 0.9131)	0.0096
ARC-Challenge	0.8419 (0.5784, 1.0000)	0.1800 (0.1646, 0.1920)	-5.792 (-6.887, -5.026)	0.6264 (0.5699, 0.7293)	0.0075
HellaSwag	0.7488 (0.4336, 0.7563)	0.2520 (0.2449, 0.2756)	-10.01 (-14.43, -9.628)	0.6357 (0.6342, 0.7124)	0.0024
Lambada	1.0000 (0.7478, 1.0000)	0.1774 (0.1505, 0.1954)	-3.977 (-4.774, -3.690)	0.3516 (0.3386, 0.4691)	0.0061
MMLU	0.9011 (0.3020, 1.0000)	0.2305 (0.2268, 0.2415)	-4.207 (-6.040, -3.954)	0.4197 (0.3755, 0.7764)	0.0022
PIQA	0.4396 (0.2719, 1.0000)	0.5683 (0.5206, 0.6025)	-4.951 (-7.662, -3.036)	0.8075 (0.4770, 0.9077)	0.0038
SciQ	1.0000 (0.2714, 1.0000)	0.02759 (3.7659e-07, 0.6920)	-1.634 (-3.935, -1.472)	2.113 (1.262, 2.162)	0.0081
TriviaQA	0.9998 (0.4204, 1.0000)	3.4328e-10 (4.9324e-26, 0.01807)	-4.385 (-7.159, -4.275)	1.492 (1.482, 1.789)	0.0053
WebQs	1.0000 (0.3478, 1.0000)	0.02107 (3.1107e-18, 0.02739)	-5.668 (-6.308, -4.309)	0.5104 (0.4521, 0.7950)	0.0071
WinoGrande	0.5039 (0.1537, 0.5154)	0.4950 (0.4852, 0.5137)	-57.76 (-153.8, -49.01)	0.2862 (0.2833, 0.3135)	0.0087
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 3%					
ARC-Easy	0.8852 (0.4598, 1.0000)	0.1498 (0.03160, 0.4605)	-3.278 (-6.289, -2.741)	0.9709 (0.7953, 1.051)	0.0083
ARC-Challenge	0.8572 (0.4645, 1.0000)	0.1709 (0.1401, 0.2321)	-5.470 (-9.441, -4.811)	0.6166 (0.5647, 0.7427)	0.0095
HellaSwag	0.7556 (0.4923, 0.7637)	0.2456 (0.2382, 0.2758)	-9.501 (-13.19, -9.210)	0.6385 (0.6375, 0.6984)	0.0022
Lambada	1.0000 (0.7284, 1.0000)	0.2017 (0.1620, 0.2287)	-3.924 (-4.896, -3.614)	0.3181 (0.2999, 0.4411)	0.0062
MMLU	0.8737 (0.3396, 0.9120)	0.2297 (0.2245, 0.2473)	-4.363 (-6.527, -4.132)	0.4603 (0.4360, 0.7552)	0.0020
PIQA	0.4379 (0.2018, 1.0000)	0.5733 (0.4199, 0.6569)	-4.576 (-10.40, -2.461)	0.7951 (0.6157, 0.8970)	0.0038
SciQ	0.2032 (0.1559, 1.0000)	0.8024 (0.4640, 0.8325)	-4.286 (-6.197, -0.8398)	0.9061 (0.09884, 1.101)	0.0077
TriviaQA	1.0000 (0.4673, 1.0000)	3.9441e-10 (1.2915e-26, 0.02202)	-3.800 (-6.027, -3.684)	1.557 (1.544, 1.856)	0.0074
WebQs	1.0000 (0.1760, 1.0000)	0.04284 (2.1853e-17, 0.05154)	-5.305 (-9.153, -4.109)	0.4699 (0.4552, 0.9828)	0.0111
WinoGrande	0.5008 (0.2141, 0.5132)	0.4980 (0.4868, 0.5146)	-63.36 (-118.8, -56.24)	0.2870 (0.2859, 0.3067)	0.0080
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 10%					
ARC-Easy	0.5973 (0.3603, 1.0000)	0.4175 (0.2657, 0.5268)	-4.606 (-8.508, -2.683)	0.7961 (0.6818, 0.8509)	0.0102
ARC-Challenge	0.8554 (0.4957, 1.0000)	0.1719 (0.1474, 0.2133)	-5.509 (-8.254, -4.763)	0.6191 (0.5671, 0.7461)	0.0077
HellaSwag	0.7686 (0.4782, 0.7814)	0.2329 (0.2209, 0.2707)	-9.204 (-13.58, -8.814)	0.6378 (0.6363, 0.7027)	0.0026
Lambada	1.0000 (0.7462, 1.0000)	0.2078 (0.1751, 0.2381)	-4.002 (-4.967, -3.747)	0.3159 (0.3027, 0.4204)	0.0060
MMLU	0.8150 (0.2729, 1.0000)	0.2407 (0.2325, 0.2492)	-5.136 (-7.202, -4.511)	0.5086 (0.4367, 0.7988)	0.0029
PIQA	0.4325 (0.2592, 1.0000)	0.5786 (0.3826, 0.6342)	-4.622 (-7.839, -2.030)	0.7883 (0.4165, 0.8748)	0.0041
SciQ	0.2774 (0.1862, 1.0000)	0.7368 (0.1267, 0.8164)	-3.241 (-4.881, -0.8278)	1.036 (0.5095, 1.965)	0.0068
TriviaQA	0.9983 (0.5154, 1.0000)	1.0469e-23 (8.9296e-29, 0.01735)	-4.015 (-6.040, -3.908)	1.525 (1.515, 1.784)	0.0080
WebQs	1.0000 (0.2636, 1.0000)	0.03153 (7.7517e-16, 0.04030)	-5.364 (-7.112, -4.443)	0.4882 (0.4753, 0.8897)	0.0100
WinoGrande	0.4989 (0.2339, 0.5116)	0.4995 (0.4897, 0.5147)	-67.61 (-122.1, -59.93)	0.2864 (0.2849, 0.3032)	0.0087
Data pool: Nemotron-CC, Filter: BETR Target-CORE Top 30%					
ARC-Easy	0.6267 (0.5370, 1.0000)	0.3909 (0.2106, 0.4662)	-4.516 (-5.633, -2.713)	0.8159 (0.6307, 0.8790)	0.0103
ARC-Challenge	0.8254 (0.6361, 1.0000)	0.1872 (0.1673, 0.2067)	-6.502 (-7.797, -5.560)	0.6432 (0.5874, 0.7043)	0.0086
HellaSwag	0.7639 (0.4908, 0.7753)	0.2373 (0.2267, 0.2677)	-9.501 (-13.31, -9.114)	0.6376 (0.6363, 0.6998)	0.0024
Lambada	1.0000 (0.7984, 1.0000)	0.1868 (0.1665, 0.2015)	-4.063 (-4.723, -3.857)	0.3451 (0.3337, 0.4367)	0.0066
MMLU	0.7986 (0.4090, 1.0000)	0.2416 (0.2350, 0.2469)	-5.488 (-6.374, -4.907)	0.5351 (0.4602, 0.7061)	0.0022
PIQA	0.4739 (0.4057, 1.0000)	0.5406 (0.2289, 0.5993)	-4.204 (-5.392, -1.901)	0.8265 (0.4495, 1.012)	0.0043
SciQ	0.2458 (0.2036, 1.0000)	0.7637 (0.4928, 0.7940)	-3.410 (-4.347, -0.8965)	0.9745 (0.09342, 1.157)	0.0075
TriviaQA	0.9966 (0.4876, 1.0000)	1.9465e-23 (2.3055e-29, 0.01879)	-4.214 (-6.792, -4.084)	1.500 (1.488, 1.767)	0.0084
WebQs	1.0000 (0.2758, 1.0000)	0.02252 (0.006499, 0.03034)	-5.567 (-8.836, -4.952)	0.5043 (0.4865, 0.8482)	0.0078
WinoGrande	0.5049 (0.2222, 0.5239)	0.4945 (0.4828, 0.5069)	-65.29 (-109.2, -57.86)	0.2865 (0.2852, 0.3054)	0.0090

Table 13: **Accuracy scaling law parameters** for BETR Target-CORE and no filtering baseline on the Nemotron-CC data pool. Each column contains the estimated value and the 95% confidence range (lower bound, upper bound). The last column shows the mean average error (MAE) for the fit evaluated on the training data.