SELECting over Tokens: Curating Pre-training Data at Scale via Token Classification

Xin Tong[*,†], **Weidong Zhang**[*,†], **Jiaang Li**, **Haibin Chen**,
Shilei Liu, **Langming Liu**, **Kangtao Lv**, **Yujin Yuan**, **Wenbo Su**, **Bo Zheng**
Future Living Lab of Alibaba
{yuheng.tx,zwd434367}@alibaba-inc.com

Abstract

The quality of pre-training data critically impacts the capabilities of large language models. Existing pipelines rely on expert-crafted heuristic rules, which primarily operate at the sample level and are based on coarse statistical indicators, thus lacking content-aware, fine-grained noise detection. While recent generative approaches, e.g., PROX-C, enable token-level refinement, their reliance on synthesizing Python code incurs prohibitive computational cost at scale and can introduce hallucinations into the refined data. To overcome these limitations, we propose **SELECting over Tokens** (**SELECT**), a novel framework that reframes data refinement as a highly efficient token classification task. SELECT classifies each token as either informative or noisy and subsequently removes the latter. This design achieves fine-grained data optimization while avoiding the inefficiency of generation, ensuring scalability. When evaluated on diverse downstream benchmarks, the model trained on SELECT-refined corpora, on average, outperforms the one trained on raw data by over 2% and exceeds the best heuristic baselines by more than 1% while preserving 17% more tokens than the latter. Furthermore, SELECT achieves higher average performance than the generative PROX-C across all experimental settings, and is 2.5x faster at inference, even with twice the parameters. Our results establish SELECT as an effective, efficient, and scalable solution for pre-training data optimization.

1 Introduction

Contemporary Large Language Models (LLMs) (Meta, 2024; Achiam et al., 2023; Anthropic, 2024; Reid et al., 2024) have demonstrated remarkable capabilities across a wide array of tasks, including creative writing (Yuan et al., 2022), programming (Li et al., 2023), and logical reasoning (Wei et al., 2022; Kojima et al., 2022). The bedrock of these achievements lies in large-scale, high-quality pre-training corpora, which endow these models with extensive world knowledge and sophisticated reasoning abilities (Together, 2023; Penedo et al., 2024).

Currently, the vast majority of pre-training corpora are sourced from the internet, a domain replete with noisy and low-quality content. This necessitates a thorough data curation process prior to training. Conventional pipelines heavily rely on expert-crafted heuristic rules (Raffel et al., 2020; Rae et al., 2021; Penedo et al., 2024), which typically make judgments at the document level based on coarse statistical metrics, rendering them content-agnostic and unable to perform fine-grained filtering. Consequently, these inflexible rules can only discard entire documents rather than surgically removing noisy tokens within them. To achieve finer granularity, some methods have shifted to the line level (Henriksson et al., 2025; Huo et al., 2025), scoring each line and discarding those below a threshold. While an improvement, these approaches still struggle with inaccurate scoring due to a lack of global context and remain too coarse, unable to perform token-level edits. Seeking even finer control, other works have pushed towards token-level refinement. Some approaches have employed medium-sized LLMs to rewrite low-quality text, aiming to preserve salient information while eliminating noise (Su et al., 2024; Maini et al., 2024). The generative nature of these methods, however, renders them computationally expensive and severely limits their scalability. Furthermore, they are prone to introducing factual hallucinations (Maini et al., 2024), which are challenging to verify. Other methods, such as PROX-C (Zhou et al., 2024) and RefineX (Bi et al., 2025), frame data refinement as a programming task, generating Python scripts to perform token-level cleaning.

[*]First two authors contributed equally and are listed in alphabetical order by last name.
[†]Corresponding authors

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 48060–48085

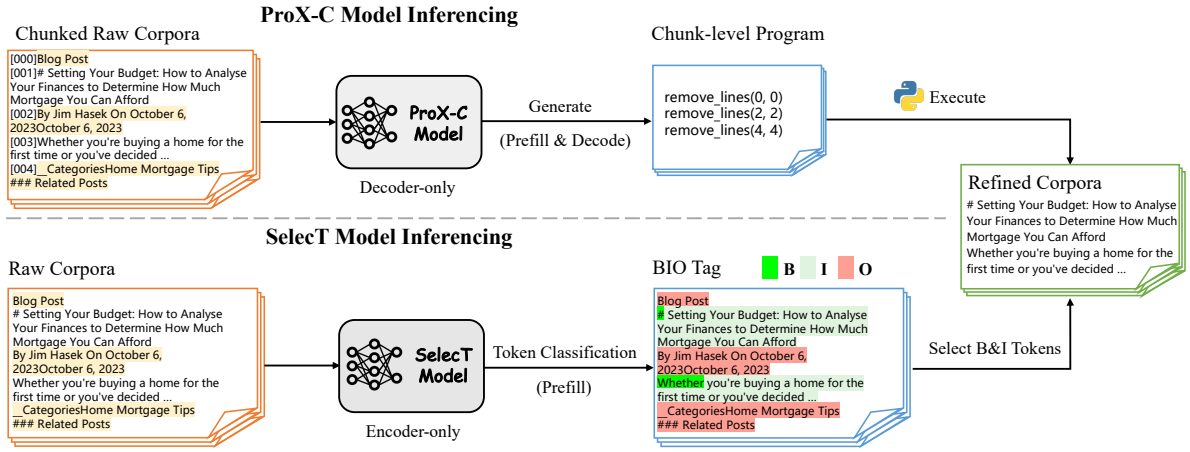

Figure 1: Comparing the inference pipelines of ProX-C (generative) and SELECT (classification). ProX-C's reliance on slow, sequential decoding creates a bottleneck, whereas SELECT's single-pass classification architecture enables significantly faster inference.

Nevertheless, ProX-C remains fundamentally generative; despite using a smaller model, it is still exceedingly time-consuming at the pre-training scale, and its generated scripts can also suffer from hallucinations, leading to erroneous text manipulations.

To address these limitations, we introduce **SELECting over Tokens** (**SELECT**), a novel framework that reformulates data refinement as a token classification problem. This reformulation results in a fundamentally different inference paradigm from generative methods like ProX-C (Zhou et al., 2024), as depicted in Figure 1. In more detail, our method (illustrated in Figure 2) begins by creating a high-quality seed dataset. We prompt a frontier LLM to refine a small set of documents under a strict, deletion-only constraint. After rigorous text alignment and verification, this process yields a corpus with token-level labels annotating content as either informative or noisy. Next, we frame this task as a sequence labeling problem (Ma and Hovy, 2016), employing the BIO tagging scheme to identify informative spans (B, I) and noisy tokens for removal (O). Finally, a small encoder-only model is fine-tuned on this labeled data to perform scalable sequence labeling by jointly modeling individual token classifications and the transition probabilities between adjacent labels.

To validate the efficacy of SELECT, we refine the raw corpus using multiple baselines (heuristics, ProX-C) and SELECT, producing a distinct 40B-token dataset for each. A 1.7B-parameter LLM is then pre-trained from scratch on each corpus. Compared with raw data, the model trained on the SELECT-cleaned corpus yields over 2% higher average scores across diverse downstream benchmarks, and exceeds the best heuristic baseline by over 1% on average while retaining 17% more tokens. Against ProX-C, SELECT consistently delivers higher average performance across all experimental settings, and—despite having twice the parameters—achieves 2.5× faster inference. Data quality analysis with the fineweb-edu classifier further shows that SELECT-processed raw corpora shift towards higher quality: the proportion of documents with scores ≥ 2 rises by 5.37%, and those scoring 1 increase by 2.50%. These results highlight SELECT as a scalable, high-performance solution for large-scale pre-training data curation.

The major contributions to our work are summarized as follows:

- We propose SELECT, a data cleaning framework that reformulates noise removal as token classification, enabling fine-grained filtering and avoiding the inefficiency and hallucinations of generative approaches.
- We design a deletion-only LLM-based seed dataset construction and verification process that produces a high-quality, token-level labeled corpus, enabling the training of token classification models.
- Experimental results show that, SELECT consistently surpasses heuristic and generative baselines (ProX-C) on average across all experimental conditions, retaining 17% more tokens than the best heuristic, delivering 2.5× faster inference than ProX-C even with twice its parameter count.

2 Approach: SELECting over Tokens

The overall workflow of the SELECT framework is illustrated in Figure 2. In the following sections,

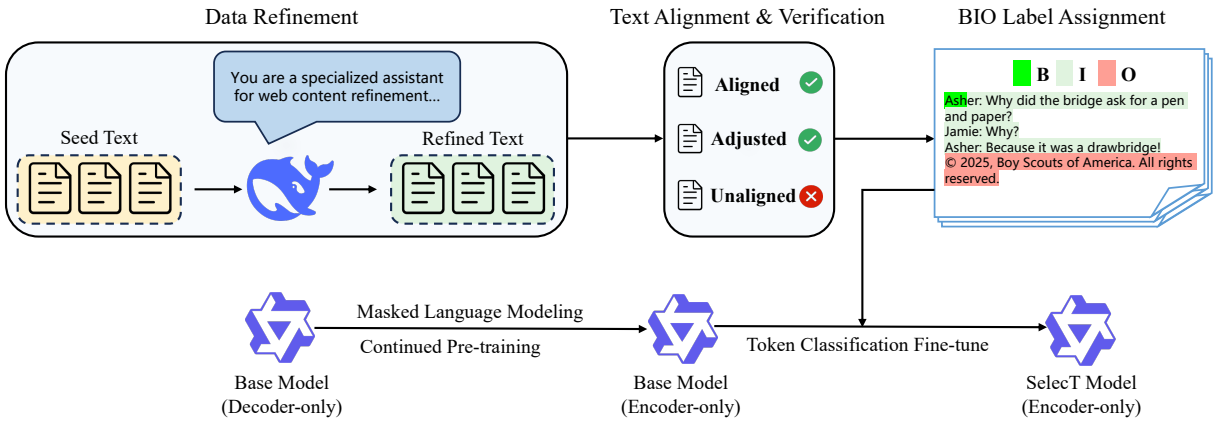

Figure 2: An overview of the SELECT Framework. We first generate a BIO-labeled dataset by using an expert LLM to refine a seed corpus under a deletion-only constraint, followed by rigorous alignment and validation. This dataset is then used to fine-tune the encoder-only base model into the final SELECT model, which is prepared by adapting a decoder-only base model via continued pre-training on a Masked Language Modeling task.

we detail its key components.

2.1 Task Definition

Given a training corpus $\mathcal{D}$, for each document d in the raw dataset, we first tokenize the text into a sequence of tokens $d = (t_1, t_2, \ldots, t_{|d|})$.

The refinement process of transforming document d into refined document $\hat{d}$ can be formulated as a token classification problem—specifically, a sequence labeling task (Ma and Hovy, 2016). Our goal is to classify each token into one of three categories following the BIO tagging scheme. The corresponding labels are:

- B: Beginning token of a main content segment
- I: Inside token of a main content segment
- O: Outside tokens representing noise content

The refined document $\hat{d}$ is then produced by preserving only the tokens labeled as B or I while removing those labeled as O.

This task formulation enables precise, token-level content refinement that preserves the original document structure by strictly removing tokens rather than rewriting them, thus avoiding hallucinations. Furthermore, the BIO tagging scheme provides a clear representation of each main content segment, imposing structural constraints on the label sequence (e.g., an I tag should follow a B or I). These dependencies are then explicitly modeled through transition probabilities to ensure sequence-level coherence (see Section 2.3 for details).

2.2 Data Construction

Annotation via LLM-driven Refinement To construct our training dataset, we sample a small-scale corpus of raw webpages and employ DeepSeek-R1 (DeepSeek-AI et al., 2025) to refine them via a carefully designed prompt (see Appendix A for the full prompt). The prompt directs the model to extract the main content (i.e., informative content) by removing noisy, boilerplate elements, operating under a critical deletion-only policy that prohibits any rewrites or paraphrasing. This ensures strict token-level alignment between the source and cleaned text, a prerequisite for our token classification task using BIO tagging.

While the prompt enumerates representative noise types (e.g., URLs, navigation) for guidance, it also empowers the model to leverage its world knowledge to more comprehensively identify and remove non-essential content. Furthermore, the LLM is instructed to discard an entire page if it lacks substantive main content or if the content is incoherent, incomplete, or exhibits low information density.

The resulting LLM output is denoted as $\hat{d} = (\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_{|\hat{d}|})$, where $\hat{t}_i$ represent the LLM-produced tokens.

While our prompt explicitly enforces a deletion-only policy, the LLM may still introduce subtle modifications. We therefore implement a rigorous verification step during the sequence alignment phase to identify and discard any samples exhibiting such alterations, ensuring only pure deletions are retained.

Token Sequence Alignment We first apply the *longest-matching-segments* algorithm (see Appendix B) to identify all contiguous character segments in d that exactly match a subsequence in $\hat{d}$ and whose length exceeds a minimum threshold

$L_{\min} = 20$ characters. Formally, this produces a sequence of aligned segment index tuples:

$$\mathcal{M} = \{(s_k, e_k, \hat{s}_k, \hat{e}_k)\}_{k=1}^{K}, \tag{1}$$

where (s_k, e_k) are the starting and ending indices in d, and $(\hat{s}_k, \hat{e}_k)$ are the corresponding indices in $\hat{d}$. All matched segments satisfy

$$e_k - s_k + 1 = \hat{e}_k - \hat{s}_k + 1 \geq L_{\min}. \tag{2}$$

The segments are ordered such that

$$\begin{aligned} &1 \leq s_1 < e_1 < s_2 < \cdots < s_K < e_K \leq |d|, \\ &1 \leq \hat{s}_1 < \hat{e}_1 < \cdots < \hat{s}_K < \hat{e}_K \leq |\hat{d}|. \end{aligned} \tag{3}$$

We categorize the alignment result into three cases:

1. **Aligned**: The matched segments completely cover $\hat{d}$ without gaps:

$$\bigcup_{k=1}^{K} [\hat{s}_k, \hat{e}_k] = [1, |\hat{d}|]. \tag{4}$$

   This case means that $\hat{d}$ is a strict filtered subsequence of d.

2. **Adjusted**: For some adjacent matched segments, the gap between them in d and $\hat{d}$ differs by at most $\Delta_{\max} = 5$ characters:

$$\begin{aligned} &|(s_{k+1} - e_k) - (\hat{s}_{k+1} - \hat{e}_k)| \leq \Delta_{\max}, \\ &\forall k \in \{1, \ldots, K-1\}. \end{aligned} \tag{5}$$

   The small discrepancy is typically due to light paraphrasing or word substitution by the LLM. In this situation, we replace the LLM-produced characters in the gap with the corresponding original characters from d, thereby merging the two matched segments into one continuous block.

3. **Unaligned**: Cases where the above two conditions are not satisfied. These alignments cannot be reliably traced back to the original sequence and are thus discarded.

BIO Label Assignment To facilitate token-level labeling, we first convert the character-level aligned spans into their corresponding token-level spans. We then generate BIO labels for each retained sample (Categories 1 and 2): the first token of a span is labeled B (Beginning), subsequent tokens within the span are I (Inside), and all tokens outside any matched span are labeled O (Outside). This yields high-quality supervised labels that faithfully reflect the LLM-cleaned main body text while guarding against inadvertent rewriting. Finally, we obtain approximately 400k high-quality training examples.

2.3 Model Training

With the BIO-labeled dataset constructed, we develop our refinement model starting from the 0.6B Qwen3-Base (Yang et al., 2025). The training process follows a two-stage paradigm: continued pre-training with Masked Language Modeling (MLM) (Devlin et al., 2019), followed by supervised fine-tuning (SFT).

First, to adapt the decoder-only model for bidirectional sequence labeling, we replace its causal attention with a bidirectional mechanism and perform continued pre-training on a 100B token FineWeb corpus using an MLM objective.

Second, the MLM-adapted model is fine-tuned on our BIO-labeled data. While a simple binary (informative/noisy) classification approach is feasible, our initial explorations revealed a common issue: such models often misclassify individual tokens within a continuous, informative text segment, leading to undesirable "chunk fragmentation". To ensure the structural integrity of the extracted content, it is crucial to model the dependencies between adjacent token labels. A standard method for this is a Conditional Random Field (CRF) layer, which motivates our adoption of BIO tagging. However, as detailed in Appendix C.2, we found the CRF layer to be numerically unstable when processing the long sequences common in our task. To overcome this, we adopt a more robust approach inspired by the core principle of CRFs: directly modeling the transition probabilities between adjacent labels.

Formally, let $\boldsymbol{x} = (x_1, \ldots, x_{|d|})$ be the sequence of final hidden states from the language model, where $x_i \in \mathbb{R}^H$ corresponds to the i-th input token. For each position $i \in \{1, \ldots, |d| - 1\}$ and any two labels $u, v \in C = \{B, I, O\}$, the transition probability from label u at position i to label v at position $i + 1$ is defined as:

$$p_i^{\text{tr}}(u \to v) = \frac{\exp\left(g_i(u \to v)\right)}{\sum_{v' \in C} \exp\left(g_i(u \to v')\right)}, \tag{6}$$

where $g_i(u \to v) = W_{u \to v} x_i + b_{u \to v}$, with learnable parameters $W_{u \to v} \in \mathbb{R}^{1 \times H}$ and $b_{u \to v} \in \mathbb{R}$.

The overall transition probability of a label sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_{|d|})$ given the hidden

states $\boldsymbol{x}$ is:

$$P_{\text{tr}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{|d|-1} p_i^{\text{tr}}(y_i \to y_{i+1}). \quad (7)$$

The total training loss $\mathcal{L}$ is a combination of the standard token classification loss and the negative log-likelihood of the transition probabilities. Formally, it is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{tr}}, \quad (8)$$

where $\mathcal{L}_{\text{cls}}$ is the average token-wise cross-entropy loss, and $\mathcal{L}_{\text{tr}}$ is the average transition loss. They are calculated as follows:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{|d|} \sum_{i=1}^{|d|} \log p_i^{\text{cls}}(y_i \mid \boldsymbol{x}) \quad (9)$$

$$\mathcal{L}_{\text{tr}} = -\frac{1}{|d|-1} \sum_{i=1}^{|d|-1} \log p_i^{\text{tr}}(y_i \to y_{i+1}). \quad (10)$$

Here, $p_i^{\text{cls}}(y_i \mid \boldsymbol{x})$ is the softmax probability of the correct label y_i for the i-th token. The final objective is to minimize $\mathcal{L}$ over the training data.

Unlike independent token-level softmax classification, our method models adjacent label dependencies via p^{tr}, yielding more consistent and structurally coherent label sequences. During inference, we adopt the Viterbi algorithm to find the optimal sequence of labels, a standard practice in traditional sequence labeling.

The rationale for choosing the base model for SELECT (e.g., why not directly use off-the-shelf encoders like BERT (Devlin et al., 2019) or ModernBERT (Warner et al., 2024)), along with further details on training and inference, is provided in Appendix C and D.

3 Experiments

3.1 Experimental Setup

Pre-training Corpus and Base Models Following the standard pipeline for constructing pre-training corpora, we begin by sampling raw HTML data from early-2025 Common Crawl[1] dumps. This data undergoes a series of standard pre-processing steps—including URL filtering, content extraction, and language identification—to produce our initial raw text corpus. The raw corpus is then cleaned using our proposed method, SELECT, as well as several baselines (heuristic rules and PROX-C), yielding a distinct 40B-token dataset for each approach. Subsequently, we pretrain a 1.7B parameter model based on the Qwen-3 (Yang et al., 2025) architecture from scratch on each of these resulting corpora.

Evaluation Tasks and Baselines In our experimental setup, we assess the performance of each pretrained model across a benchmark of ten downstream tasks, leveraging the official implementation of the LightEval framework (Fourrier et al., 2023). To ensure an equitable comparison among the different approaches, we designed our experiment around three distinct data preparation scenarios: (1) document level filtering only, (2) fine-grained refinement only, and (3) a combination of both. In the combined scenario, the fine-grained refinement process is subsequently applied to various datasets that have already undergone filtering. A comprehensive description of all filtering and refinement baselines employed in our study is provided below:

- **Document Level Filtering:** Our document level filtering baselines are categorized into two main types: rule-based and model-based. For rule-based baselines, we evaluate the individual rule sets from C4 (Raffel et al., 2020) and Gopher (Rae et al., 2021), alongside the new rules introduced with FineWeb (Penedo et al., 2024) (hereafter referred to as FineWeb rules), as well as their combination (COMB), which is the actual filtering strategy used to create the FineWeb corpus. For the model-based baseline, we employ the PROX-D model from the PROX framework.

- **Model Refine:** We use PROX-C as our primary fine-grained refinement baseline, following the 1.5k-token chunking setup from the original work. Additionally, to further validate the impact of granularity, we introduce a line-level filtering model baseline (Henriksson et al., 2025). We create two versions, hereafter referred to as Line-Filter$_{\text{SELECT}}$ and Line-Filter$_{\text{PROX-C}}$, by carefully calibrating their filtering thresholds to precisely match the respective data retention rates of the token-level methods for a fair comparison.

Further details on the pre-training and evaluation setups can be found in Appendix E and F. We selected PROX-C as the sole baseline for token-level fine-grained refinement, as the model for REFINEX

[1] https://commoncrawl.org/

Method	ARC-C	ARC-E	CSQA	HellaS	MMLU	OBQA	PIQA	SIQA	WinoG	SciQ	Avg	#Win
Raw	25.09	44.53	32.84	39.66	27.98	30.40	69.86	42.94	49.49	68.40	43.12	0 / 10
+ ProX-C	26.79	46.63	32.60	43.49	28.64	31.60	70.78	43.04	50.36	72.20	44.61	3 / 10
+ SelecT	27.39	48.65	34.56	43.48	28.66	33.20	70.67	42.84	50.67	72.80	45.29	7 / 10
Rule-based filtering: Go = Gopher rules, C4 = C4 rules, Fw = FineWeb rules, Comb = Go + C4 + Fw.												
Go	25.94	46.00	33.91	41.05	27.73	32.20	69.59	42.99	51.22	67.20	43.78	1 / 10
+ ProX-C	26.19	46.25	32.92	44.93	28.28	33.00	70.08	41.76	50.83	69.00	44.32	4 / 10
+ SelecT	26.11	47.94	35.46	44.06	27.97	33.00	70.78	41.81	51.54	69.20	44.79	5 / 10
C4	26.37	45.29	31.45	42.47	27.35	33.20	70.08	43.24	49.72	66.70	43.59	1 / 10
+ ProX-C	24.74	44.23	33.58	44.40	27.96	33.20	70.95	42.63	51.07	64.40	43.72	4 / 10
+ SelecT	26.96	46.00	34.40	44.65	27.89	33.00	70.40	43.24	50.51	66.00	44.31	5 / 10
Fw	25.85	45.37	32.35	41.53	28.31	30.80	68.72	43.14	49.80	69.90	43.58	2 / 10
+ ProX-C	25.85	45.66	34.48	44.67	28.19	31.80	69.48	42.73	49.96	71.30	44.41	3 / 10
+ SelecT	26.37	48.36	34.32	44.07	28.01	32.80	70.51	42.48	51.07	70.80	44.88	5 / 10
Comb	27.30	46.21	33.42	43.39	27.81	32.20	69.75	42.37	50.99	67.50	44.09	2 / 10
+ ProX-C	26.62	45.45	33.91	44.85	28.03	32.20	70.67	42.48	50.51	66.70	44.14	2 / 10
+ SelecT	25.17	47.05	34.23	45.64	28.02	35.00	70.13	43.35	51.70	67.40	44.77	6 / 10
LLM-based filtering: ProX-D												
ProX-D	26.96	53.91	33.99	43.85	30.50	34.40	69.91	42.43	51.38	75.00	46.23	3 / 10
+ ProX-C	29.86	53.32	36.04	45.54	30.37	35.80	69.53	42.02	50.67	74.80	46.79	3 / 10
+ SelecT	29.61	56.48	35.63	45.88	30.84	35.40	69.70	42.27	50.43	80.30	47.65	4 / 10

Table 1: Performance comparison of 1.7B models across 10 downstream benchmarks. All models are pretrained on 40B-token corpora of identical size. These corpora are created using different data curation methods: using the raw text, applying various filtering strategies, or applying an additional fine-grained refinement step to the filtered data. Within each experimental group, the best result is underlined. The **#Win** column counts the number of tasks in which a method outperformes others in its group.

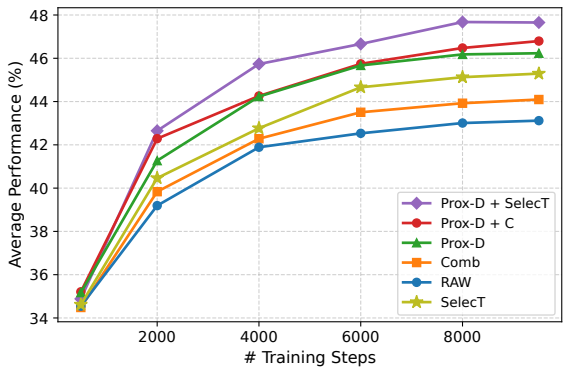

Figure 3: Downstream performance w.r.t. different training steps

was not publicly available at the time this work was conducted.

3.2 Verifying SelecT's Effectiveness

Consistent Superiority of SelecT Across Settings Following the evaluation protocol of ProX, we assess our pretrained models on ten benchmark tasks from LightEval. As shown in Table 1, when applied directly to the raw corpus, SelecT achieves an average improvement of 2.17% over the model trained solely on the raw data, and outperforms the best heuristic filtering rule—Comb (used to curate FineWeb corpora)—by 1.2% on average. SelecT also proves effective on pre-filtered corpora: when applied after heuristic filtering, it yields additional gains of 0.68–1.30%, and achieves a 1.42% improvement when applied to corpora filtered by ProX-D. Compared to the model-based refinement baseline ProX-C, SelecT consistently performs better across all experimental conditions, with average gains ranging from 0.47% to 0.86%. As illustrated in Figure 3, SelecT consistently outperforms all baselines throughout the training process. Comprehensive results for every intermediate checkpoint are available in Appendix G.1.

Notably, applying SelecT directly to the raw corpus yields superior results to all configurations where heuristic filtering is applied before SelecT. This observation underscores SelecT's high token-level precision: whereas heuristics discard entire documents that contain some noise, SelecT surgically removes only the noisy tokens, thereby preserving the valuable, informative content within those same documents. This fine-grained filtering enables more effective utilization of the training data.

To further validate this principle of granularity, we conduct a controlled comparison against a line-

Table 2: Comparison of average downstream performance for token-level versus line-level refinement methods across various corpora. Line-Filter* and Line-Filter† are Line-Filter$_{\text{SELECT}}$ and Line-Filter$_{\text{PROX-C}}$, respectively. See Appendix G.1 for per-task results.

Method	Raw	GO	C4	FW	COMB	PROX-D
SELECT	45.29	44.79	44.31	44.88	44.77	47.65
Line-Filter*	44.48	44.02	44.17	43.78	44.49	46.13
PROX-C	44.61	44.32	43.72	44.41	44.14	46.79
Line-Filter†	43.90	43.94	43.77	43.91	43.70	46.41

level model. As shown in Table 2, the token-level methods demonstrate a clear advantage over their line-level counterparts under identical data retention rates. With the sole exception of PROX-C on the C4 dataset, both SELECT and PROX-C consistently yield superior average downstream performance compared to the retention-matched Line-Filter baselines. This finding strongly confirms our central hypothesis that finer-grained data refinement is crucial for achieving optimal results, as it allows for more precise removal of low-quality content.

Efficiency Advantage over Generative Methods In terms of processing efficiency, SELECT, as a classification-based model, offers a substantial speed advantage over generative approaches like PROX-C. The performance difference stems from their fundamental architectural disparities.

The inference of a generative model like PROX-C comprises two main phases: (1) **prefill**: a parallel prompt processing phase and (2) **decode**: a sequential auto-regressive decoding phase. While the prefill is efficient, the decode process is inherently slow. It must generate tokens one by one, with each step attending to the entire input context, thus becoming a memory-bandwidth-bound bottleneck.

Table 3: Inference latency breakdown for SELECT and PROX-C.

Method	Speed (ms/sample)
PROX-C (Prefill)	6.44
PROX-C (Decode)	21.80
PROX-C (Total)	28.24
SELECT (Total)	11.23

To empirically validate this, we profile the latency of both models on a single A100-80G GPU. As detailed in Table 3, PROX-C's slow decode phase (21.80 ms/sample) accounts for over 77% of its total inference time. In contrast, SELECT's entire operation is a single parallel forward pass (11.23 ms/sample), which is architecturally analogous to only the efficient prefill stage of a generative model. By bypassing the sequential decoding bottleneck entirely, SELECT achieves a 2.5x speedup over PROX-C.

3.3 In-depth Analysis of SELECT

Impact on the Length Distribution of Original Corpora To analyze the effect of our refinement method, we examine the document length distributions before and after processing (Figure 4, Table 4). The impact is most pronounced on the raw corpus, whose original distribution is characterized by a prominent cluster of peaks corresponding to extremely short documents (under 100 tokens). While PROX-C only removes some of the most extreme outliers(e.g., <10 tokens), likely because they consist of specific noise types it targets (e.g., URLs, navigation bars), SELECT systematically suppresses the entire low-token noise region, transforming the bumpy, multi-modal distribution into a clean, unimodal one. This highlights SELECT's superior ability to generalize beyond specific, predefined noise types. Furthermore, our annotation prompt enables the complete removal of a document if it is deemed entirely noise. Consequently, SELECT aggressively prunes these short, low-quality samples (discarding 27.8% of the raw data), which explains why the average document length increases after its application.

On the pre-filtered COMB and PROX-D corpora, where this low-token noise is already absent, the distributions produced by SELECT and PROX-C become more similar in their overall shape, with both exhibiting a smoother, single-peak form. This convergence underscores that SELECT's most distinct advantage lies in its robust purification of raw, highly noisy data. Representative examples illustrating these behavioral differences are available in Appendix I.1.

Table 4: Average number of tokens for different methods.

Method	Avg. Tokens
Raw	534.45
+ PROX-C	504.91
+ SELECT	552.45
COMB	715.07
+ PROX-C	629.45
+ SELECT	643.80
PROX-D	1375.71
+ PROX-C	1233.31
+ SELECT	1079.93

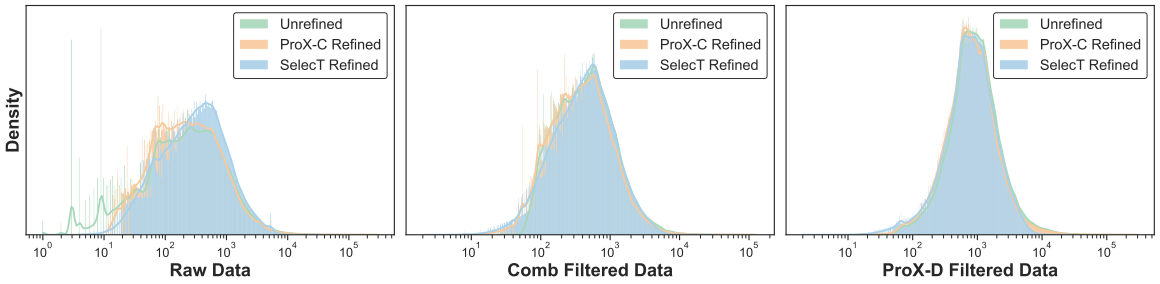

Figure 4: Token count distributions before and after applying different refinement methods to three baseline corpora: raw data, rule-based filtered data (COMB), and LLM-based filtered data (PROX-C).

Analysis of Token Retention Table 5 presents the token retention rates of various methods. When compared to coarse-grained filtering like COMB, SELECT demonstrates the advantage of its fine-grained approach. While achieving superior average downstream performance, SELECT retains 17% more tokens (59.63% vs. 50.95%) by surgically removing noise instead of discarding entire documents.

In contrast, when compared to the generative method PROX-C, SELECT retains fewer tokens (e.g., 59.63% vs. 76.89% on raw data). This disparity does not indicate lower effectiveness, but rather a more thorough cleaning process. SELECT's lower retention rate is a direct result of its stronger generalization capabilities in noise detection, which stem from the annotation process that encourages leveraging the LLM's world knowledge beyond a fixed set of noise patterns. As shown in Appendix I.1, SELECT identifies a wider variety of noise, including complex patterns like system-generated messages and content redundancy, which PROX-C often misses. This more aggressive cleaning yields a purer corpus, leading to consistently higher average downstream performance (Table 1).

Finally, in large-scale pre-training scenarios where GPU compute is the bottleneck, throughput is critical. Although SELECT retains fewer tokens per sample (approx. 77.6% of what PROX-C keeps), its 2.5x inference speedup means it produces significantly more clean data for the same computational cost. In a fixed compute budget, SELECT yields approximately 1.94 times (2.5 × 0.776) the amount of high-quality tokens compared to PROX-C, establishing it as a more scalable solution.

SELECT Boosts the Share of High-Quality Text We evaluate the quality of the raw corpus alongside the versions refined by SELECT and PROX-C, using the FineWeb-Edu classifier(Penedo et al., 2024). The results, summarized in Table 6, clearly demonstrate that both methods substantially increase the proportion of high-quality data compared to the original corpus. Notably, SELECT achieves the most impressive results. Specifically, the proportion of samples with a quality score ≥ 2 increases by 9.16% and 5.37% compared to the original data and PROX-C, respectively. Similarly, for samples with a score of 1, SELECT shows gains of 7.31% and 2.50% over the original data and PROX-C. This trend also holds on pre-filtered corpora like COMB and PROX-D (see Appendix H for details).

Table 5: Token retention ratio of different methods.

Method	Kept Ratio(%)
Raw	100
+ PROX-C	76.89
+ SELECT	59.63
COMB	50.95
PROX-D	19.38
+ PROX-C	17.33
+ SELECT	14.53

Table 6: Quality scores annotated by the FineWeb-Edu classifier.

Quality	Method	Rat.(%)
$Score = 0$	Raw	38.09
	PROX-C	29.49
	SELECT	21.62
$Score = 1$	Raw	46.55
	PROX-C	51.36
	SELECT	53.86
$Score \geq 2$	Raw	15.36
	PROX-C	19.15
	SELECT	24.52

3.4 Ablation Study

To validate the effectiveness of our full SELECT model—specifically its use of BIO tagging and transition modeling—in addressing the "chunk fragmentation" issue, we conduct an ablation study. We compare it against a simpler ablated model that employs a binary (I/O) labeling scheme and a standard token-wise classification head, foregoing any transition modeling.

The cleaning performance of both models is evaluated on our annotated test set at the token and chunk levels. For a fair comparison, predictions of B or I from our full SELECT model are both mapped to I. As shown in Table 7, both models achieve nearly identical token-level F1 scores. This is expected, as fragmentation typically stems from only a few misclassified tokens within a larger, correct segment. A more sensitive metric for this problem is the chunk-level F1 score. For this metric, analogous to NER, we treat contiguous spans of I tokens as chunks and require an exact-match for a prediction to be correct. On this challenging metric, the full SELECT model outperforms the ablated version by a substantial 12.0%, strongly confirming that our inclusion of BIO tagging and transition modeling is crucial for capturing the structural continuity of informative content, making the model more robust and less prone to omitting valid tokens within a coherent segment. We provide several representative examples in Appendix I.2 to qualitatively illustrate the behavioral differences between the two models.

Table 7: Token- and chunk-level performance of the full SELECT model versus the ablated model on the annotated test set.

Method	Precision (%)	Recall (%)	F1 (%)
token-level			
SELECT	92.3	94.4	93.3
w/o BIO & P_{tr}	91.9	94.6	93.2
chunk-level			
SELECT	51.8	47.4	49.5
w/o BIO & P_{tr}	33.1	43.3	37.5

We also investigate the impact on downstream tasks by pre-training a 1.7B model on a 40B-token corpus cleaned by the ablated model. The results, detailed in Appendix G.2, show that the full SELECT model outperforms its ablated counterpart by a modest 0.18% in average performance. This slight improvement is reasonable, as the fine-grained correction of fragmentation affects a relatively small number of tokens, and its full impact may not be apparent at our current experimental scale (1.7B model, 40B tokens). We hypothesize that this performance gap could widen on larger models, which are generally more sensitive to subtle variations in data quality. Given that our design adds negligible inference overhead, the full SELECT model remains the preferred choice for its superior structural predictions and potential for greater benefits at scale.

4 Related work

4.1 Pre-training Data Filtering

Standard data curation pipelines employ coarse, document-level filtering to prune low-quality content from web-scale corpora (Touvron et al., 2023; Together, 2023; Penedo et al., 2024). This is typically achieved using heuristic rules based on statistical indicators (Raffel et al., 2020; Rae et al., 2021; Penedo et al., 2024) or by training classifiers to score entire documents (Penedo et al., 2024; Su et al., 2024; Li et al., 2024). The primary limitation of these methods is their inability to perform fine-grained refinement, often discarding documents that contain a mix of valuable and noisy content. SELECT addresses this by operating at the token level, preserving informative segments while excising noise.

4.2 Model-based Data Refinement

To enable more granular control, recent work has focused on model-based data refinement, ranging from coarse line-level filtering (Henriksson et al., 2025; Huo et al., 2025) to finer, token-level operations. Methods like PROX-C (Zhou et al., 2024) and REFINEX (Bi et al., 2025) leverage LLMs to synthesize editing programs that perform token-level operations. However, as generative approaches, they face critical challenges: high inference latency, which limits scalability, and the risk of generating erroneous code that can corrupt the data. SELECT circumvents these issues by framing refinement as a discriminative task, offering a substantially faster and more reliable alternative.

5 Conclusion

We introduce SELECT, a novel framework that reframes data cleaning as a token classification task. On average, the model trained on SELECT-refined data outperforms one trained on raw data by over 2% and surpasses the best heuristic baseline by over 1%, while retaining 17% more tokens than the latter. SELECT also achieves higher average performance than the generative PROX-C across all settings, achieving 2.5x faster inference. Quality analysis further confirms SELECT boosts the proportion of high-quality samples. These results establish SELECT as an efficient, effective, and scalable solution for pre-training data curation.

Limitations

Our study's scope was primarily limited by significant computational constraints. This restricted our main pre-training experiments to a 1.7B parameter model on a 40B-token corpus, which in turn prevented a thorough investigation into the scaling properties of our method with larger models and data volumes. The sheer cost of each large-scale run also made it infeasible to perform significance testing across multiple random seeds to statistically validate our findings. A similar constraint applied to the continued pre-training (CT) experiments, where we were limited to the 0.6B scale Qwen3 model. We leave the exploration of these larger-scale settings and more rigorous statistical validation as critical directions for future work.

A separate limitation stems from our seed dataset generation, which relies on a single LLM (DeepSeek-R1) and a fixed prompt. This may introduce model- or prompt-specific biases. A systematic analysis of these potential effects by testing various models and prompts was beyond the scope of this work, leaving it as an important area for future investigation.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Baolong Bi, Shenghua Liu, Xingzhang Ren, Dayiheng Liu, Junyang Lin, Yiwei Wang, Lingrui Mei, Junfeng Fang, Jiafeng Guo, and Xueqi Cheng. 2025. Refinex: Learning to refine pre-training data at scale from expert-guided programs. *Preprint*, arXiv:2507.03253.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Clémentine Fourrier, Nathan Habib, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Erik Henriksson, Otto Tarkka, and Filip Ginter. 2025. Finerweb-10bt: Refining web data with llm-based line-level filtering. In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*.

Bi Huo, Bin Tu, Cheng Qin, Da Zheng, Debing Zhang, Dongjie Zhang, En Li, Fu Guo, Jian Yao, Jie Lou, Junfeng Tian, Li Hu, Ran Zhu, Shengdong Chen, Shuo Liu, Su Guang, Te Wo, Weijun Zhang, Xiaoming Shi, and 8 others. 2025. dots.llm1 technical report. *Preprint*, arXiv:2506.05767.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, and 1 others. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, and 48 others. 2023. Starcoder: may the source be with you! *Preprint*, arXiv:2305.06161.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint arXiv:2401.16380*.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

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

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

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

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Together. 2023. Redpajama: an open dataset for training large language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860.

Fan Zhou, Zengzhi Wang, Qian Liu, Junlong Li, and Pengfei Liu. 2024. Programming every example: Lifting pre-training data quality like experts at scale. *arXiv preprint arXiv:2409.17115*.

A Prompts

The prompt used in data refinement is presented in Figure 5

Prompt for Data Refinement

Task Description:
You are a specialized assistant for web content refinement. Given the raw content of a web page, your task is to remove all noisy elements and return only the valid main content.

- **Definition of Main Content:**
  The complete, substantive body of information on a page that possesses a coherent narrative structure and high informational density. Examples: article paragraphs, descriptive text, informative tables.
- **Definition of Noise:**
  Noise refers to any content that is unrelated to the main body text, including but not limited to non-essential boilerplate (e.g., headers, footers, navigation bars), meaningless URL links, advertisements, and duplicate or redundant segments. Identification of noise should be based on the contextual understanding of the original webpage, and is not confined to the aforementioned examples.

Requirements:

- **Accuracy and Fidelity:**
  You must accurately extract the main content from the source text. The original formatting of all retained text must be preserved. A brief rationale for any refinements is required.
- **Deletion-Only Constraint:**
  Only deletion operations are permitted. You are strictly forbidden from rewriting, correcting, paraphrasing, or fabricating any content. Text that carries meaningful information must not be removed.
- **Handling of Low-Quality Pages:**
  If the source text contains no substantive main content, is semantically incoherent, or exhibits very low information density, you must return an empty string in the refined_text field.

Format:

```
Input:
[doc]<raw_webpage_content>[/doc]

Output:

refinement_reason:
[doc]<brief_justification_for_refinements>[/doc]

refined_text:
[doc]<extracted_main_content_or_empty_string>[/doc]
```

Examples:
Example 1:
Input:

```
[doc]As Microsoft is expected to refresh its Surface lineup with new AI features later this year, this episode of First Ring Daily has Brad Sams
and Paul Thurrott discussing Microsoft's "peak Surface" moment.[/doc]
```

Output:

```
refinement_reason:
[doc]The text is a single, coherent, and informative sentence. It was identified as main content with no noisy elements, so nothing was removed.[/doc]

refined_text:
[doc]As Microsoft is expected to refresh its Surface lineup with new AI features later this year, this episode of First Ring Daily has Brad Sams
and Paul Thurrott discussing Microsoft's "peak Surface" moment.[/doc]
```

Example 2:
Input:

```
[doc]Posted 9th October 2020
Life can be superficial, what matters is what's on the inside!
Copyright ©2025 YogaBeamz. All rights reserved.
Website design by Adrian Burns.[/doc]
```

Output:

```
refinement_reason:
[doc]The date stamp, copyright notice, and designer credit were identified as non-essential boilerplate and removed. The central sentence was preserved
as the main substantive content.[/doc]

refined_text:
[doc]Life can be superficial, what matters is what's on the inside![/doc]
```

Example 3:
Input:

```
[doc]You should be redirected to https://www.bard-scotland.com.[/doc]
```

Output:

```
refinement_reason:
[doc]The input consists solely of a navigational redirect notice. As it contains no substantive informational content, the entire line was discarded.
[/doc]

refined_text:
[doc][/doc]
```

[Additional illustrative examples are omitted for brevity.]

ACTUAL TASK EXECUTION:
Input:

```
[doc]{input_text}[/doc]
```

Output:

Figure 5: The prompt used for data refinement

B Longest-Match Segments Extraction

The procedure for the longest-matching-segments algorithm is detailed in Algorithm 1.

C Refine Model Training Details

C.1 Base Model Selection

The selection of our base model involves a comparative evaluation of two strategies. The first strategy is to leverage an existing encoder-only model. While models like BERT and its variants are readily available, their limited 512-token context is inadequate, as it would force us to process documents in isolated chunks, disrupting long-range dependencies. We thus rule out these models and instead consider ModernBERT-large (0.4B), a more suitable candidate with its 8k-token context.

The second, alternative strategy is to convert a decoder-only model into a bidirectional encoder. We achieve this by performing continued pre-training (CT) with a Masked Language Modeling (MLM) objective. For this role, we identify Qwen3-0.6B-Base as the optimal choice, balancing performance with efficiency. It offers a 32k-token context and rich world knowledge crucial for our task, yet maintains significantly faster inference than larger models like Qwen3-1.7B-Base. After CT on 100B English tokens, this model is transformed into a powerful encoder, hereafter referred to as Qwen3-0.6B-Base-Encoder (see Appendix C.3 for training details).

To finalize our selection, we conduct a direct empirical bake-off between ModernBERT-large and Qwen3-0.6B-Base-Encoder. Using a simplified setup with binary I/O labels and a standard token-wise classification head, we fine-tune both models on our LLM-annotated dataset. As shown in Table 9, while token-level performance is comparable, Qwen3-0.6B-Base-Encoder significantly outperforms ModernBERT-large on the more challenging chunk-level F1 metric by 5.2%. This performance gap is not merely due to the parameter count difference (0.6B vs. 0.4B) but is primarily attributed to the vast disparity in their original pre-training data. The Qwen3 base model inherits knowledge from a massive 36T-token corpus, in stark contrast to the 2T tokens used for ModernBERT-large. Our 100B-token continued training served primarily as a functional adaptation to transform the model from a decoder into an encoder, rather than as the main source of its capability. This order-of-magnitude difference in initial data scale creates a fundamental knowledge gap, endowing the Qwen3-based encoder with superior generalization capabilities. This result decisively validates our choice of Qwen3-0.6B-Base as the backbone for the SELECT model.

C.2 Training Instability of the CRF Layer

During the supervised fine-tuning (SFT) stage, we initially explore the integration of a Conditional Random Field (CRF) layer to model label dependencies, a standard approach for sequence labeling (Ma and Hovy, 2016). For this setup, we adapt the Qwen3-0.6B-Base-Encoder model by stacking a head composed of two components: a score layer and a CRF layer. The score layer, implemented as a linear projection, first transforms the final hidden states of Qwen3-0.6B-Base-Encoder into token-level emission scores. Formally, let $\boldsymbol{x} = (x_1, \ldots, x_{|d|})$ be the sequence of final hidden states, where each $x_i \in \mathbb{R}^H$. For each each x_i, this layer computes a corresponding vector of emission scores $e_i \in \mathbb{R}^{|C|}$, where $|C|$ is the number of tags. Subsequently, the CRF layer takes these emission scores as input to explicitly model label dependencies by learning transition scores between successive labels.

When training the model equipped with a CRF layer using 16k and 32k context lengths, we observe severe training instability, despite applying gradient clipping with a maximum norm of 1.0. The gradient norms of both the score layer preceding the CRF and the overall model exhibit extremely frequent and large spikes. In contrast, when we limit the sequence length to 512, the model's gradient norm returns to a normal range, and spikes become infrequent and small in magnitude. As we progressively increase the context length to 2048, 4096 and 8192, we find that the frequency and magnitude of grad-norm spikes escalate with the sequence length. At 4096, the spikes are already prominent, markedly more frequent and larger than at 512. By 8192, the spikes become so frequent and large—with the largest reaching a magnitude of 1e5—that they begin to significantly disrupt training. We illustrate the evolution of the gradient norms for both the score layer and the overall model at context lengths of 512, 4096, 8192 and 16k in Figure 6. For the 8192 and 16k contexts, due to the extreme magnitude of the spikes, we plot the y-axis on a logarithmic scale and clip the maximum value at 1e20 for visualization.

We hypothesize that these large gradient spikes

Algorithm 1 Find Longest-Match Segments Between Source and Target Sequence

Require: Source character sequence: $d = (c_1, c_2, \ldots, c_{|d|})$
Target character sequence: $\hat{d} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{|\hat{d}|})$
Minimum match length: $L_{\min}$
Ensure: Match set $\mathcal{M} = \{(s_k, e_k, \hat{s}_k, \hat{e}_k)\}_{k=1}^{K}$

Initialize $\mathcal{M} \leftarrow \emptyset$
Initialize pointers $i \leftarrow 0$ for $\hat{d}$ and $j \leftarrow 0$ for d
while $i < |\hat{d}|$ **and** $j < |d|$ **do**
  Let $P \leftarrow \{\, p \mid j \le p < |d| \ \wedge\ d[p] = \hat{d}[i] \,\}$ ▷ set of start matching positions in d
  Set $\ell_{\max} \leftarrow 0$ and $p^* \leftarrow -1$
  for all $p \in P$ **do**
    $\ell \leftarrow$ LONGESTMATCHLEN($d[p:]$, $\hat{d}[i:]$)
    if $\ell > \ell_{\max}$ **then**
      $\ell_{\max} \leftarrow \ell$ and $p^* \leftarrow p$
    end if
  end for
  if $\ell_{\max} \ge L_{\min}$ **then**
    Append $(p^*, p^* + \ell_{\max},\ i,\ i + \ell_{\max})$ to $\mathcal{M}$
    $i \leftarrow i + \ell_{\max}$ ▷ advance by $\ell_{\max}$ in $\hat{d}$
    $j \leftarrow p^* + \ell_{\max}$ ▷ advance by $\ell_{\max}$ in d
  else
    $i \leftarrow i + 1$ ▷ advance by one in $\hat{d}$
  end if
end while
return $\mathcal{M}$

function LONGESTMATCHLEN($d, \hat{d}$)
  $r \leftarrow 0$
  while $r < |d|$ and $r < |\hat{d}|$ and $d[r] = \hat{d}[r]$ **do**
    $r \leftarrow r + 1$
  end while
  return r
end function

Table 8: Pretrained model architecture and training hyper-parameters.

Model	Hidden Size	Intermediate Size	Context Len	Heads	Layers	Vocab Size	# Params (w/o embed)
1.7B	2,048	6,144	4,096	16	28	151,936	1,720,574,976 (1,409,410,048)

Model	Context Length	Batch Size	Max Steps	Warmup Steps	Weight Decay	Optimizer	LR Scheduler	LR
1.7B	4,096	1,024	9,500	500	0.1	AdamW	cosine	3e-4 → 3e-6

Table 9: Token- and chunk-level performance of ModernBERT-large and Qwen3-0.6B-Base-Encoder on the annotated test set.

Method	Precision (%)	Recall (%)	F1 (%)
token-level			
ModernBERT-large	92.4	93.5	93.0
Qwen3-0.6B-Base-Encoder	91.9	94.6	93.2
chunk-level			
ModernBERT-large	27.2	39.7	32.3
Qwen3-0.6B-Base-Encoder	33.1	43.3	37.5

stem from the model's probability distribution becoming overly sharp on long sequences. In a linear-chain CRF, both the forward-backward recursion and the log-partition computation accumulate scores over the entire sequence. As the sequence length increases, the cumulative scores for different label paths diverge significantly. When processed through the softmax-like normalization in the CRF, these large score differences create an extremely sharp (low-entropy) probability distribution, where the model assigns probability mass almost exclusively to a single "winning" path. This makes the loss landscape exceptionally steep around the current parameters. Consequently, even a small discrepancy between the model's prediction and the ground truth can generate an extremely large gradient signal with respect to the emission scores. These amplified gradients then propagate back into the underlying language model, resulting in the observed grad-norm spikes for long sequences.

The inherent training instability of the CRF layer on long sequences motivates our decision to forgo its use. Instead, we introduce transition probabilities to explicitly model the dependencies between successive labels.

C.3 Two-Stage Training Procedure

Our training process involves two sequential stages: continued pre-training and supervised fine-tuning (SFT). Both stages are conducted using the FSDP framework (Zhao et al., 2023).

The first stage is **continued pre-training**. To better adapt the model for sequence labeling, which benefits from bidirectional context, we first perform continued pre-training on the Qwen3-0.6B-Base model. We sample a 100B token corpus from the FineWeb dataset (Penedo et al., 2024) for this purpose. Prior to training, we modify the model's architecture by replacing its native causal attention with bidirectional self-attention. The model is then pre-trained with a Masked Language Modeling (MLM) objective (Devlin et al., 2019). To preserve its long-context capabilities, we set the maximum sequence length to 32k tokens. We use a batch size of 512 and a cosine learning rate schedule, decaying from a peak of 5e-5 down to 1e-6.

The second stage is **supervised fine-tuning**. Following the continued pre-training stage, we fine-tune the model on our LLM-annotated BIO dataset. The SFT process runs for 6 epochs with a batch size of 128. We maintain the same 32k sequence length to handle long documents. The learning rate is again managed by a cosine decay schedule, starting at 1e-5 and decreasing to 1e-6. We select the checkpoint with the best performance on a validation set as our final refinement model.

D Model Inference

At inference time, we aim to find the most likely label sequence

$$\boldsymbol{y}^* = \arg\max_{y \in \mathcal{Y}} [\log P_{cls}(\boldsymbol{y} \mid \boldsymbol{x}) + \log P_{tr}(\boldsymbol{y} \mid \boldsymbol{x})],$$

where $\mathcal{Y}$ is the set of all possible label sequences. Given the classification probabilities P_{cls} and transition probabilities P_{tr} from the trained model, we employ the Viterbi algorithm to efficiently compute $\boldsymbol{y}^*$.

Viterbi Decoding Algorithm Let $V_i(u)$ denote the maximum log-probability of any label sequence ending at position i with label u. The recurrence relation is defined as:

$$V_1(u) = \log p_1^{cls}(u), \quad \forall u \in C, \tag{11}$$

$$V_i(u) = \max_{v \in C} \left[V_{i-1}(v) + \log p_{i-1}^{\text{tr}}(v \to u) \right] + \log p_i^{\text{cls}}(u), \quad i \geq 2. \tag{12}$$

The overall optimal score is:

$$\max_{u \in C} V_{|\boldsymbol{d}|}(u), \tag{13}$$

and the optimal label sequence $\boldsymbol{y}^*$ is then recovered by backtracking the $\arg\max$ choices at each step.

Batch-efficient Decoding Although the Viterbi algorithm involves a `for`-loop over sequence positions $i = 1, \ldots, |d|$, the per-step operations only involve small $|C| \times |C|$-shaped matrices. To maximize throughput, we first run the model forward multiple times with moderate batch sizes (determined by GPU memory constraints), accumulating the resulting p^{cls} and p^{tr} into a large batch (typically exceeding 200 samples). A single Viterbi decoding pass is then performed on the entire accumulated batch. Because the decoding computations are lightweight and scale negligibly with batch size, this strategy allows the Viterbi stage to account for less than 1% of total inference time, even for inputs up to 32K tokens in length.

E Pre-training Details

For our comparative experiments, all participating models are based on the Qwen3-1.7B architecture. We pretrain these models from scratch on a 40B token corpus using the FSDP framework; the scale of this pre-training is determined by our available computational resources. The training hyperparameters are set as follows: a batch size of 1024, a sequence length of 4096, and a learning rate with a cosine decay schedule from 3e-4 to 3e-6. Detailed model architecture and optimizer settings are summarized in Table 8.

F Experimental Evaluation Details

When evaluating the capabilities of the pre-trained model, we strictly follow the evaluation protocol of ProX, employing the same test set and the LightEval framework. The only difference is that, while ProX randomly samples 1,000 instances from each test set for evaluation, we instead use the full test set to obtain more accurate and statistically reliable results.

The complete list of evaluated datasets includes ARC-Easy and ARC-Challenge (Clark et al., 2018), CommonSenseQA (Talmor et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), and SciQ. We report normalized zero-shot accuracy as the default evaluation metric across all benchmarks.

G Evaluation Results

G.1 Main Model Evaluation Results

Full results across all downstream benchmarks for the 1.7B models are reported in Table 10 and Tables 11.

The complete experimental results comparing the token-level methods (SELECT and PROX-C) against the retention-matched, line-level filter baselines are presented in Table 12.

G.2 Ablation Study Evaluation Results

As mentioned in Section 3.4, we conducted an ablation study to measure the impact of our full refinement model (SELECT) against a simplified version. The detailed downstream performance comparison is presented in Table 13.

H Analysis of Quality Distribution on Pre-filtered Data

To demonstrate the robustness of SELECT, we analyze its effect on pre-filtered data (COMB and PROX-D) by scoring document quality with the FineWeb-Edu classifier (Penedo et al., 2024). As shown in Table 14 and Table 15, SELECT consistently improves the data quality distribution, even when used as a secondary refinement step.

On PROX-D filtered data (Table 14), SELECT achieves the highest proportion of top-quality content (Score ≥ 3) and the lowest share of low-quality content (Score ≤ 1). A similar trend is observed on COMB filtered data (Table 15), where SELECT again yields the highest percentage of high-quality documents (Score ≥ 2) and the lowest percentage of poor-quality ones (Score = 0). This confirms that SELECT's fine-grained cleaning capabilities generalize across different data quality baselines.

I Case Study

I.1 Comparison with PROX-C

This subsection presents a qualitative comparison between our proposed method, SELECT, and the

baseline, ProX-C, through several detailed case studies (Table 16 to Table 18). These cases provide a granular analysis of SelecT's superior generalization, specifically highlighting two key advantages: (1) its ability to identify a diverse spectrum of complex noise patterns that extend beyond simple boilerplate (e.g., system-generated messages, content redundancy and structured metadata), which are often missed by ProX-C; and (2) its robustness in filtering low-quality data by correctly discarding entire pages devoid of substantive content.

I.2 Ablation Case Study: Impact of BIO Tagging and Transition Modeling

To visually demonstrate the benefits of the BIO tagging scheme and transition probability modeling discussed in Section3.4, this subsection compares the outputs of the full SelecT model and its ablated counterpart. The ablated model uses a simpler binary (I/O) classification without modeling label transitions. The following cases (Table19 to Table21) illustrate how the full model's explicit modeling of label dependencies leads to more structurally coherent content extraction, preventing the fragmentation of informative chunks that can occur with the simpler ablated model.

Table 10: Full evaluation results (1/2) on a 1.7B pretrained model across different downstream tasks, with varying numbers of training steps during pretraining.

#token	ARC-C	ARC-E	CSQA	HellaSwag	MMLU	OBQA	PiQA	SIQA	WinoG	SciQ	AVG
Raw											
2000	22.87	39.77	29.16	32.27	26.34	26.40	64.47	41.76	49.88	59.00	39.19
4000	25.34	44.40	31.86	36.25	27.13	28.60	67.63	42.27	49.01	66.40	41.89
6000	23.89	43.98	32.19	38.81	27.50	28.80	68.66	42.63	49.57	69.30	42.53
8000	25.00	44.28	32.76	39.50	27.82	29.80	69.70	42.94	49.88	68.40	43.01
9500	25.09	44.53	32.84	39.66	27.98	30.40	69.86	42.94	49.49	68.40	43.12
Raw + ProX-C											
2000	22.78	41.33	28.91	34.18	26.78	28.00	64.85	41.35	49.17	62.50	39.99
4000	25.00	44.74	31.53	38.75	27.49	30.40	67.79	42.43	51.38	67.90	42.74
6000	25.94	45.71	32.35	42.09	27.83	31.00	69.86	42.73	50.36	70.10	43.80
8000	26.71	45.92	32.60	43.45	28.45	32.40	70.78	42.99	50.83	71.60	44.57
9500	26.79	46.63	32.60	43.49	28.64	31.60	70.78	43.04	50.36	72.20	44.61
Raw + SelecT											
2000	24.57	41.88	31.12	33.91	26.69	27.40	64.58	41.30	48.38	64.80	40.46
4000	24.57	45.58	32.51	39.21	27.45	30.40	68.66	41.81	50.28	67.20	42.77
6000	26.45	48.27	33.74	42.06	28.31	31.60	70.51	42.73	50.51	72.40	44.66
8000	26.88	48.78	34.64	43.37	28.65	32.60	70.29	43.09	50.36	72.60	45.13
9500	27.39	48.65	34.56	43.48	28.66	33.20	70.67	42.84	50.67	72.80	45.29
Go											
2000	23.21	41.54	28.83	33.43	26.76	29.20	64.36	42.68	49.49	60.90	40.04
4000	25.09	44.02	32.10	37.21	27.10	30.20	67.30	42.43	51.14	67.00	42.36
6000	24.49	45.37	33.33	39.98	27.39	31.60	68.72	42.99	50.67	66.40	43.09
8000	26.02	45.50	33.82	40.78	27.80	31.80	69.37	43.19	50.99	67.10	43.64
9500	25.94	46.00	33.91	41.05	27.73	32.20	69.59	42.99	51.22	67.20	43.78
Go + ProX-C											
2000	23.46	40.70	29.48	34.63	26.67	28.80	65.61	42.22	49.41	62.30	40.33
4000	25.51	44.02	31.20	39.80	27.24	30.80	67.63	42.58	50.99	64.30	42.41
6000	25.00	46.30	32.19	43.42	27.69	32.40	69.10	42.02	51.14	68.40	43.77
8000	26.11	46.13	32.76	44.60	27.78	33.00	70.18	42.12	50.83	68.50	44.20
9500	26.19	46.25	32.92	44.93	28.28	33.00	70.08	41.76	50.83	69.00	44.32
Go + SelecT											
2000	22.95	41.25	31.29	34.14	26.28	29.60	66.00	42.58	51.07	60.70	40.58
4000	25.09	45.71	33.33	40.06	27.07	30.80	68.93	42.43	50.59	66.40	43.04
6000	25.94	46.97	33.74	42.51	27.64	31.60	70.02	42.07	50.43	69.00	43.99
8000	25.77	47.90	34.97	43.79	27.83	31.60	70.78	41.81	51.22	69.00	44.47
9500	26.11	47.94	35.46	44.06	27.97	33.00	70.78	41.81	51.54	69.20	44.79
C4											
2000	24.57	39.98	29.40	34.16	26.06	29.00	64.58	42.84	49.88	62.00	40.25
4000	24.15	42.17	30.47	38.28	27.26	32.60	68.23	42.17	49.33	65.90	42.06
6000	25.51	44.57	31.20	41.03	27.15	32.00	69.31	43.19	48.86	65.90	42.87
8000	26.37	45.20	31.37	42.16	27.34	32.40	70.02	43.14	49.49	65.90	43.34
9500	26.37	45.29	31.45	42.47	27.35	33.20	70.08	43.24	49.72	66.70	43.59
C4 + ProX-C											
2000	22.53	40.15	30.88	34.76	26.70	29.80	65.61	41.25	49.49	60.80	40.20
4000	23.72	43.06	31.70	39.53	27.34	32.60	68.44	42.12	50.67	62.20	42.14
6000	25.34	44.36	32.51	42.72	28.27	32.60	69.42	41.86	50.36	65.40	43.28
8000	25.26	44.49	32.68	44.14	28.11	32.80	70.35	42.27	51.14	64.70	43.59
9500	24.74	44.23	33.58	44.40	27.96	33.20	70.95	42.63	51.07	64.40	43.72
C4 + SelecT											
2000	24.57	40.57	29.98	34.79	26.78	29.80	65.40	42.07	49.88	59.60	40.34
4000	24.23	43.73	33.66	40.31	27.35	32.20	69.48	43.14	49.96	63.80	42.79
6000	26.19	44.95	33.99	42.71	28.02	32.60	70.13	43.19	50.20	65.70	43.77
8000	26.62	45.83	34.15	44.52	27.96	33.20	70.40	43.09	50.43	65.80	44.20
9500	26.96	46.00	34.40	44.65	27.89	33.00	70.40	43.24	50.51	66.00	44.31

Table 11: Full evaluation results (2/2) on a 1.7B pretrained model across different downstream tasks, with varying numbers of training steps during pretraining.

#token	ARC-C	ARC-E	CSQA	HellaSwag	MMLU	OBQA	PiQA	SIQA	WinoG	SciQ	AVG
Fw											
2000	24.32	40.07	28.50	33.59	26.12	29.20	65.29	42.17	50.51	61.30	40.11
4000	24.66	42.59	30.88	37.43	27.20	29.60	66.27	42.53	51.38	63.80	41.63
6000	25.17	44.07	32.19	40.01	27.70	31.40	68.01	42.68	50.59	68.10	42.99
8000	26.19	45.92	32.68	41.27	28.24	31.20	68.61	43.19	49.25	71.00	43.76
9500	25.85	45.37	32.35	41.53	28.31	30.80	68.72	43.14	49.80	69.90	43.58
Fw + ProX-C											
2000	22.95	40.49	29.81	34.79	26.45	30.40	64.96	41.71	49.96	62.40	40.39
4000	25.34	44.32	32.76	39.92	27.25	30.40	68.28	42.84	49.09	66.20	42.64
6000	26.62	45.24	34.07	43.29	27.82	31.60	69.10	42.89	49.88	68.30	43.88
8000	25.26	45.66	34.48	44.45	28.33	31.00	69.42	42.27	50.59	70.70	44.22
9500	25.85	45.66	34.48	44.67	28.19	31.80	69.48	42.73	49.96	71.30	44.41
Fw + SelecT											
2000	23.98	41.20	29.89	34.32	26.46	29.00	65.51	41.86	50.67	59.90	40.28
4000	26.11	45.66	31.78	39.52	27.47	31.40	68.77	42.63	50.59	66.20	43.01
6000	26.19	46.89	33.58	42.46	27.85	33.00	70.35	42.22	50.67	69.60	44.28
8000	26.71	48.27	34.23	43.57	28.10	32.40	70.51	42.37	51.30	70.70	44.82
9500	26.37	48.36	34.32	44.07	28.01	32.80	70.51	42.48	51.07	70.80	44.88
Comb											
2000	24.66	40.87	29.16	33.85	26.45	27.80	65.13	41.50	50.75	58.20	39.84
4000	25.85	43.81	31.86	39.32	26.69	30.20	67.90	42.78	49.80	64.60	42.28
6000	26.02	45.33	33.01	41.96	27.78	32.40	69.42	42.94	50.59	65.60	43.51
8000	26.79	45.83	33.33	43.10	28.06	32.00	69.31	42.48	51.22	67.10	43.92
9500	27.30	46.21	33.42	43.39	27.81	32.20	69.75	42.37	50.99	67.50	44.09
Comb + ProX-C											
2000	24.15	39.65	29.40	35.04	26.68	27.80	66.38	41.81	50.83	60.40	40.21
4000	24.83	43.73	31.45	40.44	27.51	33.00	68.39	42.84	50.83	62.70	42.57
6000	25.51	44.61	32.02	43.28	27.57	31.80	69.59	42.37	50.59	64.20	43.16
8000	26.88	45.50	33.66	44.54	28.04	33.00	70.67	42.63	50.43	65.90	44.13
9500	26.62	45.45	33.91	44.85	28.03	32.20	70.67	42.48	50.51	66.70	44.14
Comb + SelecT											
2000	24.06	41.54	30.14	35.35	26.22	30.00	64.58	41.56	49.96	60.80	40.42
4000	24.83	44.91	33.74	40.93	26.92	32.40	69.48	42.89	51.14	64.30	43.15
6000	25.85	45.62	33.99	43.96	27.64	32.60	69.42	42.73	51.07	65.40	43.83
8000	25.09	46.30	34.07	45.32	28.07	33.40	70.08	43.24	52.25	66.90	44.47
9500	25.17	47.05	34.23	45.64	28.02	35.00	70.13	43.35	51.70	67.40	44.77
ProX-D											
2000	25.00	46.00	28.26	34.05	27.82	28.40	63.71	40.99	50.67	67.80	41.27
4000	27.73	51.01	32.19	39.24	29.43	32.40	67.57	41.40	50.75	70.60	44.23
6000	26.71	52.82	33.25	41.86	30.02	34.20	69.21	42.37	52.25	74.00	45.67
8000	27.13	54.00	33.82	43.52	30.41	34.60	69.97	42.84	50.91	74.60	46.18
9500	26.96	53.91	33.99	43.85	30.50	34.40	69.91	42.43	51.38	75.00	46.23
ProX-D + ProX-C											
2000	25.17	47.90	30.63	35.10	27.84	31.80	64.85	41.45	50.12	68.00	42.29
4000	26.62	49.75	32.51	40.44	29.07	33.40	67.30	42.48	48.62	72.40	44.26
6000	28.33	52.36	34.48	43.91	30.19	33.60	68.88	42.02	50.28	73.40	45.74
8000	29.18	53.41	35.71	45.28	30.38	35.20	69.48	42.22	49.80	74.10	46.48
9500	29.86	53.32	36.04	45.54	30.37	35.80	69.53	42.02	50.67	74.80	46.79
ProX-D + SelecT											
2000	26.71	47.52	31.53	35.36	27.78	31.00	65.13	41.97	50.12	69.40	42.65
4000	28.75	53.66	33.58	41.00	29.46	34.20	68.23	42.02	50.75	75.70	45.73
6000	29.27	55.35	34.97	44.13	30.40	33.20	69.37	42.02	50.91	77.00	46.66
8000	29.61	55.93	36.04	45.70	30.80	35.40	70.02	42.58	51.38	79.30	47.68
9500	29.61	56.48	35.63	45.88	30.84	35.40	69.70	42.27	50.43	80.30	47.65

Table 12: Full per-task results for the comparison between token-level methods and their retention-matched, line-level counterparts across all corpora.

methods	ARC-C	ARC-E	CSQA	HellaSwag	MMLU	OBQA	PiQA	SIQA	WinoG	SciQ	AVG
On Raw Corpora											
SELECT	27.39	48.65	34.56	43.48	28.66	33.20	70.67	42.84	50.67	72.80	45.29
Line-Filter$_{\text{SELECT}}$	25.94	46.84	33.99	43.20	27.99	32.60	71.22	42.27	51.54	69.20	44.48
PROX-C	26.79	46.63	32.60	43.49	28.64	31.60	70.78	43.04	50.36	72.20	44.61
Line-Filter$_{\text{PROX-C}}$	26.45	45.33	33.17	41.87	28.28	31.20	69.70	42.17	49.88	70.90	43.90
On GO filtered Corpora											
SELECT	26.11	47.94	35.46	44.06	27.97	33.00	70.78	41.81	51.54	69.20	44.79
Line-Filter$_{\text{SELECT}}$	26.71	45.62	34.48	43.42	28.40	32.20	69.70	43.60	50.59	65.50	44.02
PROX-C	26.19	46.25	32.92	44.93	28.28	33.00	70.08	41.76	50.83	69.00	44.32
Line-Filter$_{\text{PROX-C}}$	25.68	44.23	34.73	42.83	27.75	33.00	69.10	42.94	50.91	68.20	43.94
On C4 filtered Corpora											
SELECT	26.96	46.00	34.40	44.65	27.89	33.00	70.40	43.24	50.51	66.00	44.31
Line-Filter$_{\text{SELECT}}$	25.94	44.49	35.05	44.79	27.99	34.80	70.84	42.37	51.07	64.40	44.17
PROX-C	24.79	44.30	33.62	44.36	27.96	33.20	70.87	42.53	51.07	64.50	43.72
Line-Filter$_{\text{PROX-C}}$	25.60	44.57	35.05	43.71	27.71	32.40	70.24	42.68	49.33	66.40	43.77
On FW filtered Corpora											
SELECT	26.37	48.36	34.32	44.07	28.01	32.80	70.51	42.48	51.07	70.80	44.88
Line-Filter$_{\text{SELECT}}$	25.51	45.16	33.74	43.13	27.91	30.20	69.91	41.97	50.75	69.50	43.78
PROX-C	25.85	45.66	34.48	44.67	28.19	31.80	69.48	42.73	49.96	71.30	44.41
Line-Filter$_{\text{PROX-C}}$	26.19	45.79	33.25	42.79	28.00	31.20	70.51	42.22	51.14	68.00	43.91
On COMB filtered Corpora											
SELECT	25.17	47.05	34.23	45.64	28.02	35.00	70.13	43.35	51.70	67.40	44.77
Line-Filter$_{\text{SELECT}}$	27.13	45.75	34.64	45.30	28.03	33.80	70.40	43.19	51.07	65.60	44.49
PROX-C	26.62	45.45	33.91	44.85	28.03	32.20	70.67	42.48	50.51	66.70	44.14
Line-Filter$_{\text{PROX-C}}$	25.17	44.49	33.09	45.08	28.03	32.00	69.26	42.99	50.43	66.50	43.70
On PROX-D filtered Corpora											
SELECT	29.61	56.48	35.63	45.88	30.84	35.40	69.70	42.27	50.43	80.30	47.65
Line-Filter$_{\text{SELECT}}$	27.90	51.01	35.54	45.40	29.36	36.20	70.24	42.17	51.62	71.90	46.13
PROX-C	29.86	53.32	36.04	45.54	30.37	35.80	69.53	42.02	50.67	74.80	46.79
Line-Filter$_{\text{PROX-C}}$	28.07	52.19	33.42	44.31	30.47	35.40	70.46	41.86	50.28	77.60	46.41

Table 13: Downstream performance comparison between SELECT and the ablated model.

Methods	ARC-C	ARC-E	CSQA	HellaSwag	MMLU	OBQA	PiQA	SIQA	WinoG	SciQ	AVG
SELECT	27.39	48.65	34.56	43.48	28.66	33.20	70.67	42.84	50.67	72.80	45.29
w/o BIO & P_{tr}	27.13	48.82	36.86	44.34	29.06	31.60	69.48	42.02	51.54	70.20	45.11

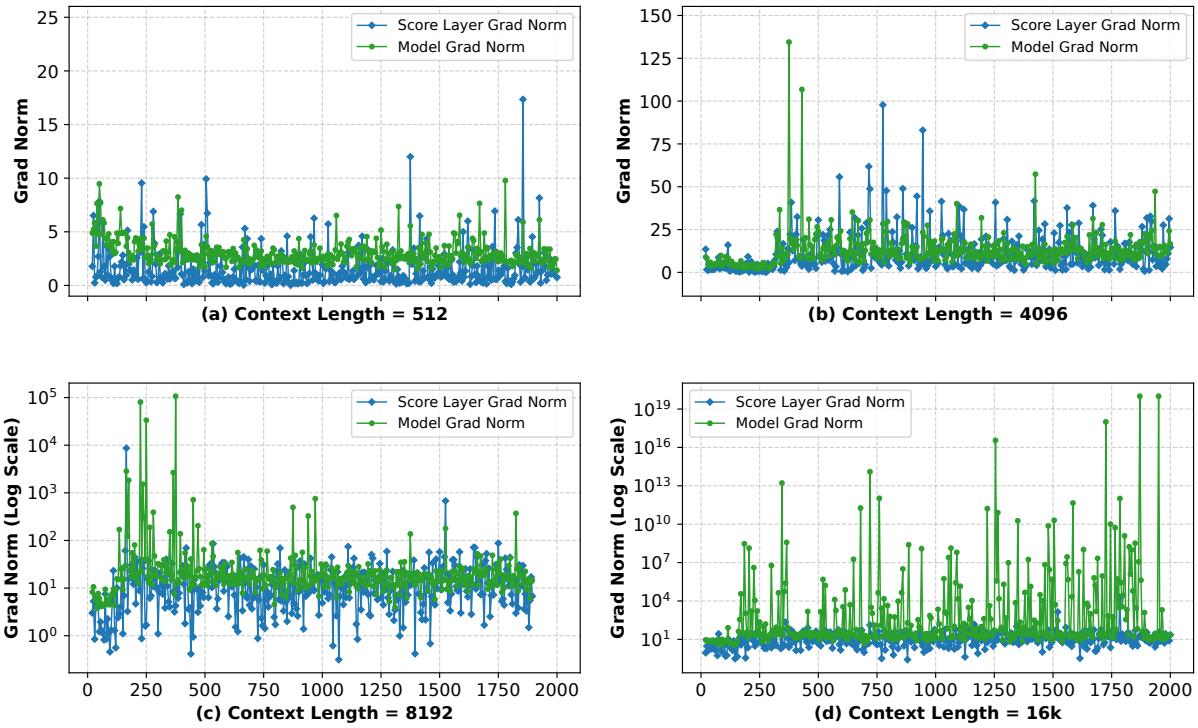

Figure 6: The gradient norm evolution over training steps under different sequence lengths.

Table 14: Distribution of quality scores on ProX-D filtered data.

Quality	Method	Rat.(%)
$Score \leq 1$	ProX-D	8.39
	ProX-C	7.94
	SELECT	7.51
$Score = 2$	ProX-D	53.41
	ProX-C	52.51
	SELECT	50.90
$Score \geq 3$	ProX-D	38.20
	ProX-C	39.56
	SELECT	41.59

Table 15: Distribution of quality scores on COMB filtered data.

Quality	Method	Rat.(%)
$Score = 0$	COMB	13.32
	ProX-C	14.02
	SELECT	12.21
$Score = 1$	COMB	60.16
	ProX-C	58.58
	SELECT	56.72
$Score \geq 2$	COMB	26.52
	ProX-C	27.40
	SELECT	31.06

Table 16: Cases (1/3) after applying ProX and SelecT. Text in red indicates content removed by each method.

Case 1
Refined by ProX:
| It is currently Sat Jan 18, 2025 12:22 am |
|
All times are UTC - 5 hours [DST] |
| Information |
|—|
Sorry but this board is currently unavailable. |
|
All times are UTC - 5 hours [DST] |
Refined by SelecT:
| It is currently Sat Jan 18, 2025 12:22 am |
|
All times are UTC - 5 hours [DST] |
| Information |
|—|
Sorry but this board is currently unavailable. |
|
All times are UTC - 5 hours [DST] |

Table 17: Cases (2/3) after applying ProX and SelecT. Text in red indicates content removed by each method.

Case 2
Refined by ProX:
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
What is Lorem Ipsum?
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Refined by SelecT:
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
What is Lorem Ipsum?
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Table 18: Cases (3/3) after applying PROX and SELECT. Text in red indicates content removed by each method.

Case 3
Refined by PROX:
2 Replies 50394 Views Last post by CFAudio Sat Jul 20, 2019 11:33 pm 3 Replies 61774 Views Last post by sonicwarrior Fri Oct 14, 2011 2:57 am 0 Replies 95966 Views Last post by Joe Malone Sat May 08, 2010 12:13 pm 0 Replies 97143 Views Last post by Joe Malone Sat Jul 04, 2009 2:58 pm 0 Replies 99915 Views Last post by Joe Malone Wed Sep 24, 2008 4:42 pm 10 Replies 126423 Views Last post by Joe Malone Tue Mar 27, 2007 7:39 pm
Refined by SELECT:
2 Replies 50394 Views Last post by CFAudio Sat Jul 20, 2019 11:33 pm 3 Replies 61774 Views Last post by sonicwarrior Fri Oct 14, 2011 2:57 am 0 Replies 95966 Views Last post by Joe Malone Sat May 08, 2010 12:13 pm 0 Replies 97143 Views Last post by Joe Malone Sat Jul 04, 2009 2:58 pm 0 Replies 99915 Views Last post by Joe Malone Wed Sep 24, 2008 4:42 pm 10 Replies 126423 Views Last post by Joe Malone Tue Mar 27, 2007 7:39 pm

Table 19: Cases (1/3) after applying SELECT and the ablated model. Text in red indicates content removed by each method.

Case 1
Refined by SELECT:
This map shows the population size and distribution of villages in Dorset after the Norman Conquest in 1086AD. The plotted data is an extract from the Open Domesday Book. Many thanks to the works of Professor John Palmer, George Slater and opendomesday.org without whose contribution this site could not have been created. Total Population(ex slaves) of the 0 Domesday Villages in 1086AD shown is 0 This would equate to a Fyrd(national service) of approximately 0 soldiers. Please Note that the estimate for the Fyrd is based on 6% of the population(excluding Priests and Slaves) being eligible for military duties. | Summary of Domesday information for this map | |—| Population Overview | |—| Occupation Overview | |—| Land Overview | |—| Animals Overview | |—| | No Animal details recorded in Domesday | Valuation | |—| | | AD1066 | AD1086 | |—| | Value(geld) | | | | |—| Hundreds population and villages Population excludes slaves | |—| | Hundred | Villages | Population | |—| |
Refined by the ablated model:
This map shows the population size and distribution of villages in Dorset after the Norman Conquest in 1086AD. The plotted data is an extract from the Open Domesday Book. Many thanks to the works of Professor John Palmer, George Slater and opendomesday.org without whose contribution this site could not have been created. Total Population(ex slaves) of the 0 Domesday Villages in 1086AD shown is 0 This would equate to a Fyrd(national service) of approximately 0 soldiers. Please Note that the estimate for the Fyrd is based on 6% of the population(excluding Priests and Slaves) being eligible for military duties. | Summary of Domesday information for this map | |—| Population Overview | |—| Occupation Overview | |—| Land Overview | |—| Animals Overview | |—| | No Animal details recorded in Domesday | Valuation | |—| | | AD1066 | AD1086 | |—| | Value(geld) | | | | |—| Hundreds population and villages Population excludes slaves | |—| | Hundred | Villages | Population | |—| |

Table 20: Cases (2/3) after applying SELECT and the ablated model. Text in red indicates content removed by each method.

Case 2
Refined by SELECT:
Ilyushin IL-2 "Sturmovik" Item Number: 1220IL-2 Sturmovik airplane model. The IL-2 was the symbol of Soviet air power in World War II. It is emblematic of the Soviet stress on ground attack aircraft as part of combined arms warfare. In 1938 the Ilyushin design bureau began work on a heavily armored low-level attack aircraft, a 'sturmovik' in Russian. This word came to be synonymous with the Il-2. Entering production in 1940, the Il-2 sported an armament of two wing-mounted armor-piercing cannon and two machine guns and carried eight rockets and 400 kilograms (880 pounds) of bombs. It had protective armor up to 12mm thick around the engine and two-man cockpit, although the rear fuselage was of plywood. The VVS (Air Forces) had 249 Il-2s in service when the war started. As aircraft factories relocated east of the Urals in 1941 to escape German bombing, Stalin characterized the Il-2 as necessary to the Red Army "like air, like bread." Production became a national priority, and the Russians built 36,163 Il-2s by November 1944, at first under appalling winter conditions as laborers erected factory walls and roofs around open-air assembly lines. Later, production rates climbed as high as 1.5 aircraft an hour at some plants, and 41,129 were built by the war's end. The first production Il-2s flew directly to frontline units before tests of the prototype were even completed. The plane's easy handling, powerful armament, and invulnerability to ground fire made it a devastating ground attack aircraft, especially with the tenacity of desperate pilots, and the Germans called it the "Black Death." But losses were extremely high from German fighters, even after a rear gun was added for self defense-14,200 were claimed downed in 1943 and 1944 alone. The Luftwaffe even formed specially-trained fighter units to target Il-2s, and several of Germany's highest-ranking aces gained most of their kills against Il-2s. Mahogany Wood. Scale: 1/32. Wingspan 18 inches, Length 14 3/8 inches.
Refined by the ablated model:
Ilyushin IL-2 "Sturmovik" Item Number: 1220IL-2 Sturmovik airplane model. The IL-2 was the symbol of Soviet air power in World War II. It is emblematic of the Soviet stress on ground attack aircraft as part of combined arms warfare. In 1938 the Ilyushin design bureau began work on a heavily armored low-level attack aircraft, a 'sturmovik' in Russian. This word came to be synonymous with the Il-2. Entering production in 1940, the Il-2 sported an armament of two wing-mounted armor-piercing cannon and two machine guns and carried eight rockets and 400 kilograms (880 pounds) of bombs. It had protective armor up to 12mm thick around the engine and two-man cockpit, although the rear fuselage was of plywood. The VVS (Air Forces) had 249 Il-2s in service when the war started. As aircraft factories relocated east of the Urals in 1941 to escape German bombing, Stalin characterized the Il-2 as necessary to the Red Army "like air, like bread." Production became a national priority, and the Russians built 36,163 Il-2s by November 1944, at first under appalling winter conditions as laborers erected factory walls and roofs around open-air assembly lines. Later, production rates climbed as high as 1.5 aircraft an hour at some plants, and 41,129 were built by the war's end. The first production Il-2s flew directly to frontline units before tests of the prototype were even completed. The plane's easy handling, powerful armament, and invulnerability to ground fire made it a devastating ground attack aircraft, especially with the tenacity of desperate pilots, and the Germans called it the "Black Death." But losses were extremely high from German fighters, even after a rear gun was added for self defense-14,200 were claimed downed in 1943 and 1944 alone. The Luftwaffe even formed specially-trained fighter units to target Il-2s, and several of Germany's highest-ranking aces gained most of their kills against Il-2s. Mahogany Wood. Scale: 1/32. Wingspan 18 inches, Length 14 3/8 inches.

Table 21: Cases (3/3) after applying SELECT and the ablated model. Text in red indicates content removed by each method.

Case 3
Refined by SELECT:
The BBC and the government have put together daily lessons for children across the UK. If you have finished the work we have set or you fancy something different, take a look. Oak National Academy I'm very pleased to welcome Oak National Academy, a new online classroom and resource hub, which has launched to support teachers until schools re-open. The online classroom will provide free lessons and resources for pupils from reception through to year 10. All resources, lessons and more information can be accessed here. As part of a collective response to the crisis, 40 teachers will be producing over 180 free online lessons each week, which will cover a range of subjects from English and maths to arts and languages. BBC Bitesize This week the BBC is launching its own education package as part of their Charter across TV and online – helping to keep children learning and supporting parents. You can find more information about this here. The key difference between the BBC offer and the Oak National Academy is that the Academy is a planned and sequenced series of lessons for schools and trusts to use as part of their planned curriculum offer. Whereas the BBC offer is aimed more at parents and pupils. Please note that we are not responsible for any of this content. If there are any technical issues, please contact the BBC or Oak Academy directly. We have found that the Oak Academy videos work best on Internet Explorer (the sound doesn't always work on Chrome, but we are sure they will be ironing out these teething issues moving forward).
Refined by the ablated model:
The BBC and the government have put together daily lessons for children across the UK. If you have finished the work we have set or you fancy something different, take a look. Oak National Academy I'm very pleased to welcome Oak National Academy, a new online classroom and resource hub, which has launched to support teachers until schools re-open. The online classroom will provide free lessons and resources for pupils from reception through to year 10. All resources, lessons and more information can be accessed here. As part of a collective response to the crisis, 40 teachers will be producing over 180 free online lessons each week, which will cover a range of subjects from English and maths to arts and languages. BBC Bitesize This week the BBC is launching its own education package as part of their Charter across TV and online – helping to keep children learning and supporting parents. You can find more information about this here. The key difference between the BBC offer and the Oak National Academy is that the Academy is a planned and sequenced series of lessons for schools and trusts to use as part of their planned curriculum offer. Whereas the BBC offer is aimed more at parents and pupils. Please note that we are not responsible for any of this content. If there are any technical issues, please contact the BBC or Oak Academy directly. We have found that the Oak Academy videos work best on Internet Explorer (the sound doesn't always work on Chrome, but we are sure they will be ironing out these teething issues moving forward).