Accelerating Large Language Model Pretraining via LFR Pedagogy: Learn, Focus, and Review

Neha Prakriya, Jui-Nan Yen, Cho-Jui Hsieh, Jason Cong
University of California, Los Angeles
{nehaprakriya, juinanyen, chohsieh, cong}@cs.ucla.edu

Abstract

We introduce an effective and scalable data selection technique to accelerate the pretraining of large language models (LLMs). Given the variation in quality and informativeness of web-scale corpora, we present the Learn-Focus-Review (LFR) paradigm-a dynamic training approach that adapts to the model's learning progress. Inspired by human learning techniques like spaced repetition, LFR tracks the model's learning performance across data instances and prioritizes revisiting challenging and diverse regions of the dataset that are more prone to being forgotten, enabling better retention and more efficient learning. Through experiments spanning over 2200 GPU hours, we show that LFR significantly enhances data efficiency in pretraining while improving downstream performance across commonsense reasoning, question answering, problem-solving, language modeling, and translation tasks. LFR consistently achieves lower perplexity and higher accuracy using just 5%–19% of the training tokens as models trained on the full dataset. Notably, LFR matches the performance of industry-standard Pythia models with up to 2× the parameter count while requiring only 3.2% of the training tokens. Unlike prior work on data selection, LFR models are Chinchilla-optimal demonstrating the effectiveness of our training methodology.

1 Introduction

LLMs have achieved remarkable success in understanding and generating human language. This success is driven by the ever-increasing model parameter sizes which require web-scale training datasets like SlimPajama (Soboleva et al., 2023), CommonCrawl (Penedo et al., 2023; Raffel et al., 2023), Pile (Gao et al., 2020), and OpenWebText (Radford et al., 2019; ope), leading to unsustainable training costs. Between 2016 and 2023, model training costs have skyrocketed by a factor of 750× every two years (Gholami et al., 2024), while GPU memory has scaled at a much slower pace of 2× every two years. For example, pretraining the GPT-4 model (OpenAI et al., 2024) was estimated to have cost around $100M USD over a period of 3-4 months using 25k NVIDIA A100 GPUs (gpt).

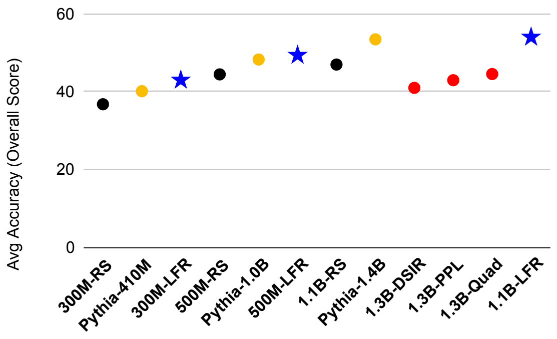


Figure 1: Average accuracy norm across commonsense reasoning, problem-solving, world knowledge, and reading comprehension tasks. Across model sizes (300M–1.1B), LFR (stars) outperforms full-dataset training (RS in black circles) by 6%, Pythia (yellow circles) by 1.5%, and Quad (Zhang et al., 2024) (red circle) by 9%, using only 3–6% and 65% of the training tokens of Pythia and Quad, respectively. Notably, Pythia and Quad have larger parameter counts. See Section 5 for details.

As such, a key challenge for unlocking the next generation of language models is to significantly reduce training costs while retaining or improving downstream task performance.

Data quality and selection play a key role in the development of cost-effective and high-performance models (Hoffmann et al., 2022; Brown et al., 2020; Tirumala et al., 2023; Abbas et al., 2023; lla, 2024). In fact, DeepSeek-V3 technical report (DeepSeek-AI et al., 2025) and the Llama 3.1 Technical Report (lla, 2024) highlight the importance of data quality through curated data mixes and sophisticated data preprocessing pipelines to minimize redundancy and maximize corpus diversity.

Recent work on data selection for pretraining has achieved great strides in reducing the overall training time. Methods like D4 (Tirumala et al., 2023), SemDeDup (Abbas et al., 2023), MiniPile (Kaddour, 2023; min), DSIR (Xie et al., 2023), and perplexity-based filtering (Marion et al., 2023; Chen et al., 2024; Muennighoff et al., 2023) rely on similarity metrics, clustering, or perplexity to filter data. However, data importance evolves throughout training and depends on model architecture, making static filtering inherently limited in effectiveness. While (Zhang et al., 2024) employ a dynamic data selection approach using the multi-armed bandit technique, they select 30B tokens from the SlimPajama dataset to train a 1.3B parameter model. However, according to the Chinchilla scaling laws (Hoffmann et al., 2022), this token count exceeds the optimal range for models of this size, suggesting that their selected subsets may contain redundant or lower-quality data. Other studies propose leveraging state-of-the-art (SOTA) pretrained LLMs like GPT-4 (Wettig et al., 2024) or proxy models, as seen in MATES (Yu et al., 2024) and RHO-1 (Lin et al., 2024), to assess data quality for a target model. However, these approaches rely on existing separately trained models, which may introduce a mismatch between the data needed for optimal convergence and the data selected.

We address the high training cost of LLMs and the shortcomings of existing data selection methods by drawing inspiration from spaced repetition (Smolen et al., 2016a; spa). This scientifically proven technique enhances retention by strategically presenting information at optimal intervals, ensuring that the most relevant data is introduced at the right time for efficient learning. Building on this foundation, we propose the *Learn-Focus-Review* (LFR) training paradigm. Figure 1 displays the overall efficacy of LFR. Our work offers the following contributions:

1. Profile LLM pretraining to observe multiple descent behavior in 25-78% of the training tokens from web-scale corpuses, which are forgotten multiple times during training.

2. Develop a Learn-Focus-Review (LFR) training pipeline that dynamically gauges the LLM's learning pace, focusing on complex data blocks while regularly reviewing all data blocks to prevent forgetting.

3. Conduct over 2200 GPU hours of training experiments using the AMD MI250, AMD MI210, and AMD MI100 GPUs. We pretrain Llama and GPT models of varying sizes from scratch on the SlimPajama (627B) and OpenWebText (9B) datasets and evaluate them on several downstream tasks from the commonsense reasoning, question-answering, problem solving, language modeling, and translation domains.

4. LFR results in significantly lower perplexity and higher accuracy compared to baseline models trained on the full dataset, achieving these improvements by training on just 5-19% of the training tokens used by the baseline. All our models are Chinchilla-optimal.

5. LFR outperforms the performance on 70% of tasks of the Pythia models with up to 2$\times$ the parameter count while requiring only 3-6% of the training tokens.

6. LFR outperforms prior state-of-the-art data selection work by 9-13% in downstream task accuracy while using only 65% of the training tokens.

7. Observe that LLMs first learn conversational and anecdotal data, before being able to retain factual, instructional, and coding language information in long-term memory.

In the following sections, we examine prior works on efficient LLM pretraining before diving deeper into our proposed training strategies and design decisions.

2 Related Work

Prior works on efficient pretraining of LLMs using data selection have primarily focused on using distance metrics and clustering techniques. Tirumala et al. (2023) proposes D4, which deduplicates and selects cluster centers in the embedding space generated by pretrained models. SemDeDup (Abbas et al., 2023) prunes semantic duplicates using pretrained models. It can successfully prune 50% of the training data with minimal performance loss. MiniPile (Kaddour, 2023; min) uses the pretrained E5-Large (Wang et al., 2024) model to embed documents in the Pile dataset and clusters them to select a smaller pretraining corpus of $\sim$6GB. DSIR (Xie et al., 2023) proposes selecting subsets from large unlabeled datasets through importance resampling to match the distribution of the target dataset. However, considering the high cost of training, it is unsustainable to sample a new subset and pretrain the LLM from scratch for every new downstream task.

More recently, perplexity-based and influence function-based filtering techniques have been proposed (Marion et al., 2023; Lin et al., 2024; Muennighoff et al., 2023; Chen et al., 2024; Wettig et al., 2024; Yu et al., 2024), which use proxy models to obtain perplexity/influence scores for different data points and assess sample importance. However, these methods require an additional pretrained model, increasing computational overhead. Moreover, if the proxy model has a different architecture from the target model, its assessment of data importance may not accurately transfer, leading to suboptimal data selection and inefficiencies in training.

The Chinchilla scaling laws (Hoffmann et al., 2022) derive an optimal model size–to–training tokens ratio for fixed compute budgets, finding that parameters and data should scale roughly 1:1. We observe that several of the prior works discussed in this Section do not incorporate Chinchilla scaling laws (Hoffmann et al., 2022) into their data selection strategies, leading to suboptimal filtering of web-scale corpora and potential overtraining. For example, Zhang et al. (2024) present Quad, a data selection method which calculates influence scores to measure a data point's impact on model performance. They select 30B tokens from the SlimPajama dataset (627B) for their 1.3B model and continual pretraining of the 7B model. This indicates that the models have been overtrained or trained on redundant tokens.

3 Problem Formulation and Profiling

3.1 LLM Pretraining Objective

Given an LLM model parameterized by weights θ and a web-scale dataset D, we first tokenize all documents in the dataset and obtain context-length-sized sequences of tokens, called data blocks, s_i such that the training corpus becomes $D = \{s_1, s_2, s_3, ...s_n\}$. For the SlimPajama and OpenWebText datasets used in this paper, the context length is 1024 tokens, with a total of 627B and 9B tokens, respectively. Given one such sequence of tokens or data block, $s_i = \{x_1, x_2, ...x_n\}$, the training objective is to autoregressively predict the next M tokens:

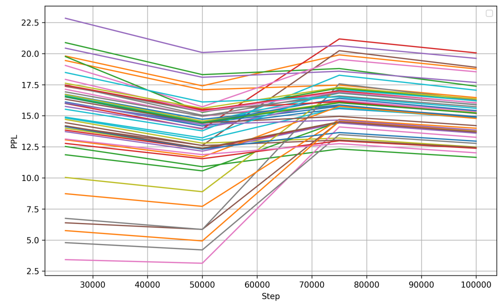


Figure 2: PPL trajectories of data samples from the SlimPajama dataset as processed by the Llama-300M model, focusing on a subset of 50 samples for clarity. Notably, 78.5% of the samples exhibit this behavior, characterized by multiple descent patterns rather than a steady decline. This indicates that the model frequently forgets and relearns data during training, highlighting inefficiencies in traditional training dynamics

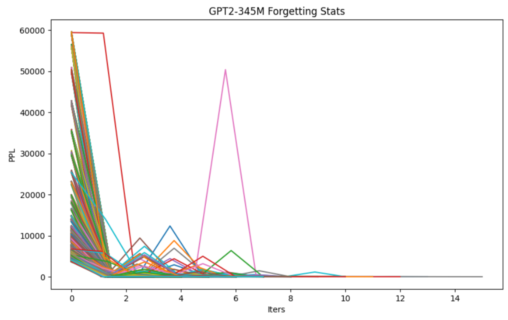


Figure 3: PPLs of data samples being forgotton by the GPT2-345M model on the OpenWebText dataset. This multi-descent behavior is exhibited by 20% of the data.

$$p_\theta(y \mid x) = \prod_{i=1}^{M} p_\theta(y_i \mid y_{1:i-1}, x). \quad (1)$$

3.2 Observations from Training on Randomly Sampled Data

In order to better understand the drawbacks of this traditional training technique, we profile the pretraining process for the Llama and GPT models. The training hyperparameters and model configurations are provided in the Appendix A. Similarly to Marion et al. (2023), we use *perplexity* as a metric to monitor the training progress. Given a token sequence $s_i = \{x_1, x_2, ..., x_n\}$ from the dataset D, perplexity is computed as:

$$PPL(s_i) = \exp\left(\frac{1}{|s_i|}\sum_{x_j \in s_i} NLL(x_j)\right), \quad (2)$$

where $NLL(x_j)$ is the negative log likelihood of token x_j computed as follows:

$$NLL(x_j) = -\log P(x_j \mid x_{<j}; \theta). \qquad (3)$$

Using this metric, models exhibiting lower perplexities are considered better since this indicates a high probability of selecting text closest to the raw dataset.

The observed PPL values associated with each data block are classified as one of the following:

1. *Learned*: recorded perplexities monotonically decrease.

2. *Unlearned*: recorded perplexities monotonically increase.

3. *Forgotten*: recorded perplexities first increase and then decrease. Such an upward and downward trend may repeat any number of times during training.

Based on this classification, we observe that 78.5% of the data blocks are forgotten at least once in the Llama model (Figure 2), compared to 25% in the GPT model (Figure 3). We hypothesize that more data blocks are frequently forgotten in the Llama model due to the higher complexity and challenge posed by the SlimPajama dataset, as opposed to the OpenWebText dataset. It is important to note that the SlimPajama dataset is an aggregation of seven datasets, including sources such as GitHub, Wikipedia, and CommonCrawl. In fact, of the data blocks that are forgotten, 82% are forgotten multiple times during training, i.e., they display *multiple descent behavior* (Figure 3). Xia et al. (2022) reported a double-descent behavior for the OPT models (Zhang et al., 2022), and our above experiment further demonstrates that the "forgetting" can happen multiple times in LLM pretraining.

4 LFR Training Methodology

Based on our profiling observations in Section 3.2 we propose to replace traditional autoregressive language modeling methods with Spaced Repetition (Tabibian et al., 2019). Spaced Repetition is an evidence-based learning method proven to improve information retention and learning pace in humans (Smolen et al., 2016b). In this technique, challenging pieces of information are reviewed more often, at regular intervals, and easier pieces of information are presented to the learner less often. Our algorithm is detailed in Algorithm 1. We pretrain our models with a combination of the following three steps:

1. **Learn**: Initially, we present the model with the entire dataset and train on randomly selected data blocks for p_1 steps, as normally seen in the traditional approach (line 1 in Alg. 1). p_1 can be configured by the user based on the available compute budget, model, and dataset. In single-epoch training (lines 3-7 in Alg. 1), we measure the perplexities (PPLs) of all data samples in the training set and cluster the data embeddings (inputs to the model's last layer). For multi-epoch training (lines 8-11 in Alg 1), we record the perplexities for all data blocks during the p_1 steps. Depending on the training style (single or multi-epoch), we either pass the clustered embeddings and PPL values or the PPL values observed during training to the next step. The following two phases can be repeated up to $reps$ times, depending on the available compute budget.

2. **Focus**: We provide two variations of the Focus stage based on the number of training epochs.

   (a) Single-epoch training: We discard s_1% of the clusters (line 6 in Alg 1). Within the retained clusters, we perform weighted sampling from sub-clusters, prioritizing regions of the retained clusters which the model finds most challenging (line 7 in Alg. 1). Sub-clusters with higher PPL are assigned greater sampling weights, enabling a hierarchical focus on the most critical regions. For instance, during Llama training, GitHub code emerged as the most challenging cluster. Within this cluster, the Focus stage further emphasizes sampling from C++ code, which proved more difficult for the model, over Python code. In this phase of training, we restrict the weighted sampling of data points to this reduced subset for p_2 steps. s_1 and p_2 are user-controlled hyperparameters.

   (b) Multi-epoch training: We discard s_1% of the data blocks (line 10 in Alg. 1) with the lowest PPL values. In doing so, we provide a mechanism for shifting the model's focus towards learning data blocks that were determined to be difficult.

3. **Review**: Next, we reintroduce all of the removed data blocks and train the model by randomly sampling from the entire corpus for p_3 steps (line 13 in Alg. 1). This ensures that we allow the model to review and revisit data blocks which it may have forgotten.

Algorithm 1 LFR Training Methodology

```
Require: Training dataset $D$, model $M$ with initial parameters $\theta_0$, hyperparameters $p_1$, $s_1$, $p_2$, $p_3$, $reps$, and $epochs$.
Ensure: Minimization of Equation 3.
 1: $PPLs, \theta_{p_1} \leftarrow$ Learn($\theta_0, D, p_1$)
 2: for $r = 1, 2, \ldots, reps$ do
 3:     if $epochs == 1$ then
 4:         $D_k \leftarrow Cluster(D)$
 5:         $Sort(PPLs, D_k)$
 6:         $S_{sub} \leftarrow (1 - s_1) \times D_k$
 7:         $S_1 \leftarrow sample(S_{sub}, PPLs)$
 8:     else
 9:         $Sort(PPLs, D)$
10:         $S_1 \leftarrow (1 - s_1) \times D$
11:     end if
12:     $\theta_{p_2} \leftarrow$ Focus($\theta_{p_1}, S_1, p_2$)
13:     $PPLs, \theta_{p_3} \leftarrow$ Review($\theta_{p_2}, D, p_3$)
14: end for
    $Return\ \theta$
```

Our training strategy is simple, intuitive and human-like. It gives the model an opportunity to learn from all of the data, prioritize and relearn forgotten data points, and review data samples from harder regions of the dataset more frequently than they would have been using random sampling. While the static clustering-based techniques (Tirumala et al., 2023; Abbas et al., 2023; Kaddour, 2023) presented in Section 2 allow for accelerated training, they are not designed to suit the multi-descent training dynamics observed in Section 4 and require pretrained model embeddings to calculate distance metrics. Furthermore, prior methods including perplexity-based pruning methods (Marion et al., 2023) are static. Sections 5.4 and the Appendix characterize the data blocks found easy and hard by the LLM, and demonstrate why static, clustering-based data selection methods achieve poor downstream task performance. Lastly, our approach does not require any pretrained models to obtain embeddings. Our focused training strategy allows the model to absorb harder information (data blocks with higher perplexity) faster, by presenting them more number of times.

5 Evaluation

In this section, we present a comprehensive evaluation of LFR. We pretrain the Llama models of sizes 300M, 500M, and 1.1B and the GPT models (Radford et al., 2019) of various sizes between 124M and 1.5B parameters. We use the SlimPajama (Soboleva et al., 2023) (627B) and OpenWebText dataset (ope) (9B) and train from scratch using 4 AMD MI100, 4 AMD MI210 GPUs, and 8 AMD MI250 GPUs. Our pretraining experiments utilize a fully sharded data parallel (FSDP) approach. All model configurations and training hyperparameters of our experiments are detailed in the Appendix A.

Our models and all baselines are evaluated across a diverse set of downstream tasks spanning multiple domains: (1) Commonsense reasoning (HellaSwag, Winogrande, PIQA), (2) General knowledge (ARC_C, ARC_E, MMLU, Natural Questions), (3) Reading comprehension (OpenbookQA, BoolQ), (4) Language modeling (WikiText-2, WikiText-103, LAMBADA, 1BW), and (5) Translation (WMT-14). Performance results and comparisons to prior state-of-the-art methods are detailed in Sections 5.3.

Section 5.4 analyzes the impact of the Focus and Review stages and the data LFR prioritizes in SlimPajama. The Appendix provides examples, details on retained/dropped data across models, evidence that LLMs learn instructions and code after facts and anecdotes, and a sensitivity study on LFR hyperparameters.

5.1 LFR Configuration

We pretrain the Llama models for 100k steps, using 9.8B tokens for the 300M and 500M models and 19.6B tokens for the 1.1B model, following the Chinchilla scaling law (Hoffmann et al., 2022) to ensure optimal data utilization. First, we Learn for 20k steps ($p_1 = 20k$). Next, we cluster the data and discard 57.2% of the clusters, retaining only the 3 most challenging clusters out of 7 based on their PPL values ($s_1 = 50$). We chose this configuration based on our limited pretraining budget and profiling in Section 3.2, which showed that 78.5% and 25% of data samples are forgotten at least once during training for the Llama and GPT models, respectively. We then apply the Focus

Model	Tokens	Arc_C	Arc_E	Boolq	HS	OBQA	Piqa	WG	Avg
300M-RS	50B	17.29	39.06	33.17	32.3	28.83	58.36	48.54	36.79
Pythia-410M	300B	20.1	**44**	40	**35.82**	29.59	61.8	49.7	40.14
300M-LFR	9.8B	**23.61**	39.52	**54.86**	35.44	**30.56**	**63.21**	**53.88**	**43.01**
500M-RS	50B	25.1	43.7	53.7	36.5	32.6	65.1	52.2	44.47
Pythia-1.0B	300B	27.05	48.99	**60.83**	47.16	**31.4**	**69.21**	53.43	48.29
500M-LFR	9.8B	**28.11**	**52.89**	58.72	**50.65**	31.1	68.66	**55.72**	**49.4**
1.1B-RS	50B	27.31	50.27	60.58	38.11	31.11	66.67	54.99	47
Pythia-1.4B	300B	30.1	61.7	62.11	**55.18**	30.2	72	**63.1**	53.48
DSIR	30B	20.14	49.28	61.41	30.89	16.2	61.17	47.99	41.01
PPL	30B	20.82	45.41	58.35	35.92	18.8	66.89	54.62	42.97
1.3B-Quad	30B	20.99	52.27	62.14	34.41	20.00	70.04	52.09	44.56
1.1B-LFR	19.6B	**29.18**	**63.47**	**62.23**	54.27	**34.89**	**73.29**	61.12	**54.06**

Table 1: Zero-shot performance (acc_norm for all except Winogrande and Boolq which use acc) on downstream tasks evaluated using LLM Evaluation Harness (Gao et al., 2024). RS refers to the random sampling baseline, HS refers to HellaSwag, and WG refers to Winogrande. The model with the highest performance (measured by acc_norm) is highlighted in bold. Notably, LFR models are trained using only 3.2-6% of the tokens required to train Pythia models of comparable size, yet they achieve higher accuracy in 70% of cases. Additionally, LFR models consistently outperform the random sampling baseline by a large margin, despite being trained on 19.6% of the pretraining tokens.

stage for 60k steps ($p_2 = 60k$), prioritizing the retained high-PPL clusters. It takes <10min to cluster which can be hidden by the high training latency. We provide a detailed analysis on the hierarchical clustering and the data points found easy and difficult in Section 5. Lastly, we Review the entire dataset for the last 20k steps ($p_3 = 20k$). In the case of the GPT models, we Learn for 1 epoch ($p_1 = 1$), Focus on 50% of the data for 1 epoch ($s_1 = 50, p_2 = 1$), Review the entire dataset for another epoch ($p_3 = 1$), and Focus on 30% of the data for 5 epochs ($reps = 2, s_2 = 70, p_4 = 1$). This configuration is chosen based on the findings in Section 3.2. Figure 3 reveals that forgotten samples are typically forgotten multiple times, requiring an average of 4 presentations to be learned. For GPT, we use the first three phases to identify these samples and allocate 5 epochs focusing on 30% of them in the final phase to ensure long-term retention.

These configurations are tunable based on the available pretraining budget and the optimal tokens estimated through the Chinchilla scaling laws. Furthermore, we test LFR's sensitivity to hyperparameters p_1, s_1, p_2, p_3, and $reps$ in the Appendix A.

5.2 Baselines

We evaluate the models pretrained using LFR with a comprehensive set of prior works and industry-standard checkpoints. They include:

1. Industry-standard models: We compare the Llama models trained through LFR with Pythia models (Biderman et al., 2023) of up to $2\times$ the size obtained from EleutherAI's Huggingface[1]. These models have been trained on 300B tokens while the LFR models were trained on 9.8B-19.6B tokens (3.2-6% of the tokens). We compare the GPT models pretrained through LFR for 40k iterations with the same GPT architectures pretrained by OpenAI [2] for 800k iterations. We use the same batch size as these models (Refer to the Appendix for details) by adjusting the gradient accumulation steps and the per-device batch size.

2. Random Sampling: while the previous baselines ensures that we compare with industry-standard models, we also train and compare LFR against the same models pretrained using random sampling with $5.10\times$ and $20\times$ more tokens than LFR for the Llama and GPT models respectively. This baseline allows us to test LFR against the same models trained through traditional autoregressive techniques.

3. Prior works: We compare our training methodology with the models trained through the

[1]https://huggingface.co/models?other=pythia
[2]https://huggingface.co/openai-community

current state-of-the-art data selection methods such as Quad (Zhang et al., 2024), static-PPL based filtering (Marion et al., 2023), DSIR (Xie et al., 2023), and MiniPile (Kaddour, 2023) in Section 5.3.

5.3 Performance on Downstream Tasks

We evaluate Llama models trained with LFR on commonsense reasoning, general knowledge QA, and reading comprehension, comparing accuracy norms with baselines in Table 1. LFR outperforms random sampling (RS) by 6% while using 2.4×–5× fewer training tokens and improves accuracy over Pythia by 1.5% despite using only 3.2–6% of the tokens. Compared to prior SOTA data selection, LFR achieves greater dataset pruning while improving downstream performance. Notably, their models are over-trained per Chinchilla laws, highlighting suboptimal data selection.

We test the GPT models on language modeling tasks and compare with the OpenAI baseline in Table 2 by measuring the PPL. Note that our models are trained on 5% of the training tokens as compared with the OpenAI models, further validating that data quality is more important than quantity. We find that the PPL reduction obtained by LFR increase as the dataset size increases (from WikiText-2 to 1BW). Also, smaller models show a larger PPL reduction by using LFR than larger models. On average, using our approach, perplexity was reduced by 4.92, 3.26, 2.17, and 1.40 for the GPT 124M, 345M, 774M, and 1.5B models, respectively.

We also test the LFR-trained models on standard benchmarks from the translation, question-answering, world knowledge, and problem solving domains in Table 3. LFR models trained with 20× fewer training iterations achieves better performance than models trained using random sampling. Details of each of the datasets is provided in the Appendix A.

5.4 Ablation Study

In this section, our goal is to understand the impacts of the Focus and Review stages of LFR and exploring more aggressive data selection strategies by varying the hyperparameters p_1, s_1, p_2, p_3, and $reps$.

5.4.1 Impact of Focus

Consider training the Llama 300M parameter model on the SlimPajama dataset, which comprises

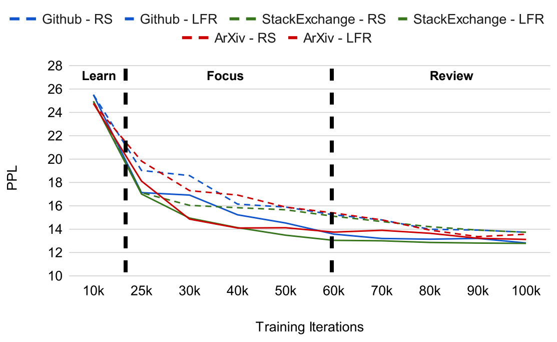


Figure 4: PPL values are tracked at different training iterations for the clusters identified as challenging and prioritized during the Focus stage of LFR. The dotted line represents the PPL values for the same clusters when trained with random sampling (RS). Notably, LFR facilitates accelerated learning of these challenging data points between 20k and 60k iterations (the Focus stage), whereas random sampling consistently results in higher PPL values throughout.

of seven sub-datasets sourced from CommonCrawl, Github, C4, Books, Wikipedia, StackExchange, and ArXiv. During the Focus stage, LFR employs weighted sampling from the three most challenging clusters while discarding clusters with the lowest perplexity (PPL). Additionally, within the retained clusters, LFR performs hierarchical sampling by prioritizing regions with higher PPL, further refining the data selection process. LFR classifies the Github, StackExchange, and ArXiv clusters as more challenging at 20k iterations, than the other four data sources.

Figure 4 illustrates the training dynamics of challenging data points. LFR (solid line) accelerates learning of these harder examples compared to random sampling (dotted line), ensuring complex information is learned earlier, which drives the performance gains in Table 1. In the Review stage, discarded clusters (CommonCrawl, C4, Books, Wikipedia) are reintroduced, bringing LFR and random sampling closer together. However, LFR retains the benefits of the Focus stage by performing marginally better on the challenging sections.

5.4.2 Impact of Review

Next, we analyze the impact of the Review phase on data points deemed simple and discarded during Focus. Unlike prior data selection methods, LFR reintroduces these samples, preventing catastrophic forgetting. Figure 5 highlights the importance of Review by plotting PPL values for easy data points under LFR (solid line) and random sampling (dot-

Model	WikiText-2	WikiText-103	LAMBADA	1BW
124M-OpenAI (800k iters)	22.1	31.58	18	39.18
124M-RS (40k iters)	23.32	23.42	17.71	39.49
124M-LFR (40k iters)	19.81	22.49	16.61	32.27
345M-OpenAI (800k iters)	19.82	22.05	14.26	29.95
345M-RS (40k iters)	21.11	21.8	14.84	30.66
345M-LFR (40k iters)	16.31	17.48	13.7	25.52
774M-OpenAI (800k iters)	15.93	18.53	13.74	26.52
774M-RS (40k iters)	16.71	18.89	14.10	28.56
774M-LFR (40k iters)	15.11	14.58	12.51	23.83
1.5B-OpenAI (800k iters)	13.80	16.59	12.15	23.87
1.5B-LFR (40k iters)	13.10	14.37	11.23	22.09

Table 2: PPL results for language modeling datasets across model sizes. Here, N-OpenAI refers to the OpenAI baseline (trained for 800k iterations), N-RS refers to the random sampling baseline (trained for 40k iterations), and N-LFR refers to our proposed training pedagogy (trained for 40k iterations), where N is the number of model parameters.

Model	Iters	WMT (BLEU)	NQ (Acc)	MMLU				
				STEM (Acc)	HM (Acc)	SS (Acc)	Other (Acc)	Avg. (Acc)
1.5B OpenAI	800k	11.5	4.1	24.5	24.8	**24.0**	**27.8**	25.3
1.5B LFR	40k	**11.8**	**4.61**	**26.1**	**27.2**	23.8	25.1	**25.5**

Table 3: LFR-trained GPT models evaluated on translation (WMT-14 (wmt)), question-answering (Natural Questions (Kwiatkowski et al., 2019)), and world knowledge and problem solving (MMLU (Hendrycks et al., 2021) domains using the BLEU scores and accuracy metrics. Note that NQ refers to Natural Questions, HM refers to Humanities, SS refers to Social Sciences, Other refers to business, health, and other miscellaneous topics, and Avg. refers to the average accuracy across all 57 subjects in MMLU. We compare our 1.5B parameter model with those trained by OpenAI for 20× more training iterations. The model with the superior performance is highlighted in bold.

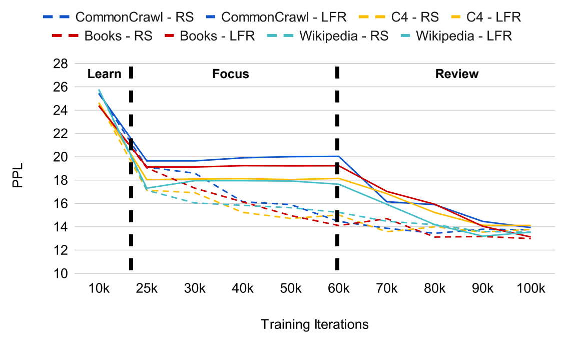


Figure 5: PPL values are tracked at different training iterations for the clusters identified as easy, discarded during the Focus stage, and reintroduced during the Review phase. The dotted line represents the PPL values for the same clusters when trained with random sampling (RS). Notably, we demonstrate that models forget the data points discarded during training, unless reintroduced to the training corpus as in the case of LFR.

ted line). During Focus, when the model prioritizes challenging clusters like GitHub, StackExchange, and ArXiv (Figure 4), it forgets discarded data (solid line rises above dotted). The Review phase restores these points, ensuring better model performance and giving LFR a distinct edge over other methods (Section 5.3). See the Appendix for raw examples of easy and difficult samples identified by LFR.

5.5 Overall Learning Schedule

LFR reveals that models follow a structured learning trajectory: first mastering conversational and anecdotal data (CommonCrawl, C4, books), then retaining factual knowledge (Wikipedia), and finally learning code, QA, and scientific content (ArXiv). By recognizing this progression automatically as shown in Sections 5.4.1 and 5.4.2, LFR optimizes training by dynamically guiding the model at its own learning pace, ensuring efficient and targeted learning.

6 Conclusion

We introduced LFR (Learn-Focus-Review), a novel data selection paradigm that accelerates LLM pre-training while significantly reducing training costs. Through 2200 GPU hours of experiments, LFR achieved lower perplexity and higher accuracy while using up to 20× fewer training iterations than traditional methods. Our findings show that LLMs follow a natural learning progression—first acquiring conversational data, then factual knowledge, and finally mastering code and scientific concepts. By dynamically guiding learning, LFR provides a scalable, cost-effective alternative to existing pre-training strategies. We hope this work inspires further research into more adaptive and efficient training paradigms.

7 Acknowledgments

This work is partially supported by the PRISM Center under the JUMP 2.0 Program, CDSC Industry Partners (https://cdsc.ucla.edu/partners/), AMD HACC Program, and NSF 2048280, 2331966, 2325121, 2244760. The authors would like to thank AMD for their generous support provided through the AMD AI & HPC Fund Research Award.

8 Limitations and Ethical Considerations

LFR presents the following directions for future work:

1. LFR is evaluated on models up to 1.5B parameters using open-source corpus like the SlimPajama dataset, constrained by our compute resources. With the clear success on models of such scale, we hope to inspire researchers to validate such focused learning approaches for different model families, and domains.

2. The sensitivity study in Section 5.4 and the Appendix reveals that the hyperparameters selected in Section 4 have a large impact on the performance of the trained model. Due to our limited compute budget, we are unable to present more comprehensive hyperparameter tuning experiments than those presented in Section 5.4.

References

a. Arc Challenge Dataset. https://huggingface.co/datasets/allenai/ai2_arc.

b. Arc Easy Dataset. https://huggingface.co/datasets/allenai/ai2_arc.

a. BookCorpus Dataset. https://huggingface.co/datasets/bookcorpus/bookcorpus.

b. BoolQ Dataset. https://huggingface.co/datasets/google/boolq.

GPT-4 Cost Estimation. https://en.wikipedia.org/wiki/GPT-4#:~:text=Sam%20Altman%20stated%20that%20the,was%20more%20than%20%24100%20million.

HellaSwag Dataset. https://huggingface.co/datasets/DatologyAI/hellaswag.

MiniPile Dataset. https://huggingface.co/datasets/JeanKaddour/minipile.

OpenBookQA Dataset. https://huggingface.co/datasets/allenai/openbookqa.

OpenWebText Dataset. https://huggingface.co/datasets/Skylion007/openwebtext.

PIQA Dataset. https://huggingface.co/datasets/ybisk/piqa.

Spaced Repetition: Wikipedia. https://en.wikipedia.org/wiki/Spaced_repetition.

WikiText Dataset. https://huggingface.co/datasets/Salesforce/.

WinoGrande Dataset. https://huggingface.co/datasets/allenai/winogrande.

WMT-14 Hugging Face Dataset. https://huggingface.co/datasets/wmt/wmt14.

2024. Meta Llama 3. https://ai.meta.com/blog/meta-llama-3/.

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *Preprint*, arXiv:2303.09540.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *Preprint*, arXiv:1312.3005.

Jie Chen, Zhipeng Chen, Jiapeng Wang, Kun Zhou, Yutao Zhu, Jinhao Jiang, Yingqian Min, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, Yankai Lin, Ruihua Song, Jun Xu, Xu Chen, Rui Yan, Zhewei Wei, Di Hu, Wenbing Huang, and Ji-Rong Wen. 2024. Towards effective and efficient continual pre-training of large language models. *Preprint*, arXiv:2407.18743.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *Preprint*, arXiv:2101.00027.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. 2024. AI and Memory Wall. *Preprint*, arXiv:2403.14123.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Jean Kaddour. 2023. The MiniPile Challenge for Data-Efficient Language Models. *Preprint*, arXiv:2304.08442.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, yelong shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2024. Not all tokens are what you need for pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When Less is More: Investigating Data Pruning for Pretraining LLMs at Scale. *Preprint*, arXiv:2309.04564.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. Scaling data-constrained language models. *ArXiv*, abs/2305.16264.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *Preprint*, arXiv:2306.01116.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Paul Smolen, Yili Zhang, and John Byrne. 2016a. The right time to learn: mechanisms and optimization of spaced learning. *Nature Reviews Neuroscience*, 17.

Paul Smolen, Yili Zhang, and John H. Byrne. 2016b. The right time to learn: mechanisms and optimization of spaced learning. *Nature Reviews Neuroscience*, 17(2):77–88.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Behzad Tabibian, Utkarsh Upadhyay, Abir De, Ali Zarezade, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. 2019. Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences*, 116(10):3988–3993.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S. Morcos. 2023. D4: Improving LLM Pretraining via Document De-Duplication and Diversification. *Preprint*, arXiv:2308.12284.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. Qurating: selecting high-quality data for training language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Ves Stoyanov. 2022. Training trajectories of language models across scales. *ArXiv*, abs/2212.09803.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. *Preprint*, arXiv:2302.03169.

Zichun Yu, Spandan Das, and Chenyan Xiong. 2024. Mates: Model-aware data selection for efficient pretraining with data influence models. *ArXiv*, abs/2406.06046.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ye Yuan, Guoren Wang, and Conghui He. 2024. Harnessing diversity for important data selection in pretraining large language models. *ArXiv*, abs/2409.16986.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

A Appendix

A.1 Experiment Details

Datasets The datasets used for our experiments are detailed below:

1. ARC-Challenge (arc, a): A subset of the AI2 Reasoning Challenge with 2,590 challenging multiple-choice science questions designed to test advanced reasoning and knowledge.

2. ARC-Easy (arc, b): A subset of the AI2 Reasoning Challenge with 5,117 relatively easier multiple-choice science questions focusing on basic reasoning and recall.

3. BoolQ (boo, b): A dataset of 16,000+ boolean (yes/no) questions paired with passages, requiring models to infer answers from supporting evidence.

4. HellaSwag (hel): A dataset with 70,000+ multiple-choice questions focused on commonsense reasoning and contextual understanding, particularly in describing scenarios.

5. OpenBookQA (Ope): A multiple-choice question-answering dataset with 5,957 questions requiring knowledge retrieval from a science "open book" and commonsense reasoning.

6. PIQA (Piq): A physical commonsense reasoning dataset with 20,000+ binary-choice questions about everyday situations and physical interactions.

7. Winogrande (win): A dataset with 44,000+ sentence pairs designed to test commonsense reasoning through pronoun disambiguation challenges.

8. WikiText (wik): the WikiText language modeling dataset consists of 100M tokens extracted from Wikipedia articles with high rating. It features two different variants, namely, WikiText-2 and WikiText-103 which differ in the number of tokens and vocabulary size. WikiText-2 consists of 2M tokens and a vocabulary size of 33k whereas WikiText-103 is larger with 103M tokens and a vocabulary size of 267k.

9. LAMBADA (Paperno et al., 2016): the LAMBADA dataset is extracted from the BookCorpus dataset (boo, a) and contains 10k passages. This dataset is useful for testing the ability of an LLM to capture long-range dependencies in text. The objective of this model is to predict the final word in a set of sentences, where humans need at least 50 tokens of context to accurately anticipate the word.

10. One Billion Word Benchmark (Chelba et al., 2014) (1BW): this dataset contains one billion words extracted from the WMT 2011 News Crawl data and is used to measure progress in statistical language modeling.

11. WMT-14 French-English Translation (Artetxe et al., 2018): This dataset contains 36 million training sentence pairs for english to french translation. The test set, which is used for evaluation purposes, consists of 3,003 sentence pairs.

12. Natural Questions (Kwiatkowski et al., 2019): This dataset contains question-answer pairs from Google Search and Wikipedia-based annotations. The training, validation, and test sets consist of 307,372, 7,830, and 7,842 examples.

Models: Tables 4 and 5 describes the different model configurations and pretraining hyperparameters used in LFR for the Llama models.

300M	500M	1.1B	
Layers	12	11	22
#Heads	16	32	32
n_embd	1024	2048	2048

Table 4: Number of layers, attention heads, and the embedding dimensions in the Llama models used for pretraining.

Tables 6 and 7 describes the different model configurations and pretraining hyperparameters used in LFR for the GPT-2 models.

Pretraining: Table 7 shows the hyperparameters for pretraining the GPT-2 124M-1.5B parameter models.

Note that OpenAI pretrained the GPT-2 models using a batch size of 512. Due to insufficient GPU memory, we adjust the number of gradient accumulation steps to achieve the same effective batch size of 512.

Finetuning: We use all the same hyperparameters as pretraining, except for the following:

1. Learning rate: 3.00E-5

Parameter	Value
Context Length	1024
Embedding Dimension	(768, 1024, 2048)
Total Iterations	100,000
Effective Batch Size	768
Block Size	4096
Weight Decay	1.00E-1
Adam β_1	0.90
Adam β_2	0.95
Warmup Iterations	8000
Minimum Learning Rate	4.00E-5
Maximum Learning Rate	4.00E-04
Learning Rate Schedule	Cosine Decay
Learning Rate Decay Iterations	100,000
GPUs	(4x AMD MI210, 4x AMD MI210, 8x AMD MI250)

Table 5: Pretraining hyperparameters for the Llama 300M-1.1B parameter models. Parameters with multiple values (e.g. Embedding dimensions, batch size, gradient accumulation steps, and GPUs) specified in brackets are for the 300M, 500M, and 1.1B parameter models respectively.

2. Learning rate schedule: Constant
3. Total iterations: 50

A.2 Limitations and Ethical Considerations

LFR presents the following directions for future work:

1. LFR is evaluated on models up to 1.5B parameters using web-scale datasets such as SlimPajama, constrained by our compute resources. With the clear success on models of such scale, we hope to inspire researchers to validate such focused learning approaches for different model families, and domains.

2. The sensitivity study in this Appendix reveals that the hyperparameters selected in the evaluation section have a large impact on the performance of the trained model. Due to our limited compute budget, we are unable to present more comprehensive hyperparameter tuning experiments than those presented later in this Appendix.

	124M	355M	774M	1.5B
Layers	12	24	36	48
#Heads	12	16	20	25
n_embd	768	1024	1280	1600

Table 6: Number of layers, attention heads, and the embedding dimensions in the GPT-2 models used for pretraining.

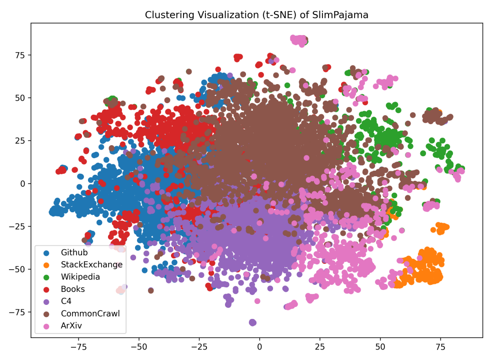


Figure 6: Clustering the data embeddings from the SlimPajama dataset using the Llama-300M model at the 50k training step.

A.3 Llama Pretraining - Data Importance

In this section, we study the data points identified as easy and challenging by LFR when pretraining with the SlimPajama dataset. Listing A.3 provides an example of a code snippet from Github classified as easy by LFR, and discarded in the Focus stage of the Llama model training. Listing A.3 provides an example of a data sample retained from the Github cluster. Note that this code is more complex, presents an opportunity to the model to improve its coding capabilities as opposed to the variable declarations in Listing A.3.

Listing 1: Code snippet classified as easy by LFR, primarily consisting of variable declarations. As seen from the code, it contributes minimally to enhancing the model's coding capabilities.

```
package frclibj;

import edu.wpi.first.wpilibj.
    Timer;

public class TrcDbgTrace
{
    public static final String
        ESC_PREFIX        = "\u001b
        [";
```

Parameter	Value
Context Length	1024
Embedding Dimension	(768, 1024, 1280, 1600)
Total Iterations	40000
Effective Batch Size	512
Batch Size	(16, 16, 8, 4)
Gradient Accumulation Steps	(32, 32, 64, 128)
Block Size	1024
Weight Decay	1.00E-01
Adam β_1	0.9
Adam β_2	0.95
Warmup Iterations	2000
Minimum Learning Rate	6.00E-05
Maximum Learning Rate	6.00E-04
Learning Rate Schedule	Linear
Learning Rate Decay Iterations	40000
GPUs	(4xMI100, 4xMI210, 4xMI210, 4xMI210)

Table 7: Pretraining hyperparameters for the GPT-2 124M-1.5B parameter models. Parameters with multiple values (e.g. Embedding dimensions, batch size, gradient accumulation steps, and GPUs) specified in brackets are for the 124M, 345M, 774M, and 1.5B parameter models respectively.

```
    public static final String
        ESC_SUFFIX          = "m";
    public static final String
        ESC_SEP             = ";";

    public static final String
        SGR_RESET           = "0";
    public static final String
        SGR_BRIGHT          = "1";
    public static final String
        SGR_DIM             = "2";
    public static final String
        SGR_ITALIC          = "3";
    public static final String
        SGR_UNDERLINE       = "4";
    public static final String
        SGR_BLINKSLOW       = "5";
    public static final String
        SGR_BLINKFAST       = "6";
    public static final String
        SGR_REVERSE         = "7";
    public static final String
        SGR_HIDDEN          = "8";
    public static final String
        SGR_CROSSEDOUT      = "9";

    public static final String
        ESC_NORMAL          =
        ESC_PREFIX;
}
```

Listing 2: Code snippet classified as challenging by LFR. This code consists of a function which executes an Oracle query and returns a scalar value. As seen from the code, it contributes significantly to enhancing the model's coding capabilities as compared with Listing A.3.

```
/// <summary>
/// Executes an Oracle query that
    returns a single scalar value
    as the result.
/// </summary>
/// <param name="commandText">The
    Oracle query to execute</
   param>
/// <param name="parameters">
   Optional parameters to pass to
    the query</param>
/// <returns>The result of the
   query as an object</returns>
public object QueryValue(string
   commandText, IEnumerable
   parameters)
{
    object result;

    if (String.IsNullOrEmpty(
        commandText))
    {
        throw new
            ArgumentException("
            Command text cannot be
             null or empty.");
    }

    try
    {
        ensureConnectionOpen();
        var command =
            createCommand(
            commandText,
            parameters);
        result = command.
            ExecuteScalar();
    }
    finally
    {
        ensureConnectionClosed();
    }

    return result;
}
```

Similarly, we also provide examples of question-answer pairs from StackExchange which were discarded and retained in the Focus stage of the Llama pretraining in Listings A.3 and A.3 respectively.

Listing 3: Question-answer pair from StackExchange classified as easy by LFR. The question revolves around a process in PayPal which does not contribute as much to the answering capability or world knowledge of the model.

```
Q: PayPal IPN $_POST['txn_id']
    not set. I'm using the PayPal
    sandbox to do a subscribe
    button, and then when I get
    the IPN response for a
    subscription or a subscription
     cancellation $_POST['txn_id']
     is never set.
So I don't know how to identify
    transactions to only accept
    unique ones.
Thanks!
EDIT: for example, all the info
    that I have in POST for a
    subscr_cancel are:
amount1, amount3, address_status,
     subscr_date, payer_id,
    address_street, mc_amount1,
    mc_amount3, charset,
    address_zip, first_name,
    reattempt,
    address_country_code,
    address_name, notify_version,
    subscr_id, custom,
    payer_status, business,
    address_country, address_city,
     verify_sign, payer_email,
    btn_id, last_name,
    address_state, receiver_email,
     recurring, txn_type,
    item_name, mc_currency,
    residence_country, test_ipn,
    period1, period3,
    correlation_id.

A: According to Table 2. Summary
    of subscription variables:
For subscription variables, the
    transaction ID (txn_id) is
    only available for USD Payment
     and Multi-Currency Payment
    transaction types (txn_type).

As expected, PayPal will not send
     the txn_id to your IPN for
    the transaction type,
    subscr_cancel, and will only
    send txn_id if the transaction
     type is subscr_payment.

For further explanation on which
    variables are sent to your IPN
     URL based on your transaction
    , please check out IPN and PDT
     Variables.

Have you checked $_REQUEST['
    txn_id'] as this may be sent
    to your server via GET.
```

Listing 4: Question-answer pair from StackExchange classified as challenging by LFR. The question revolves around solving an ODE which contributes more to the learning of the model than Listing A.3.

```
Q: Passing additional iteration-
    dependent inputs to ode45
I'm trying to solve a
    differential equation using
    the ode45 function. Consider
    the following code,
[t1,X2] = ode45(@(t,x)fun(t,x,C1,
    C2,C3,C4),t0,X01);

where parameters C1, C2, C3, and
    C4 are column vectors, which
    should be available to the
    function that ode45 is
    referring to (fun.m).
I want the values to change after
     every iteration, so for
    example, at the beginning the
    entry of C1 I want is C1(1),
    in the next iteration it's C1
    (2), etc.
How can I implement that?

A: You may have noticed that the
    official docs are not too
    helpful in this scenario (as
    they pretty much force you to
    use global variables - which
    is doable, but discouraged).
 Instead, I'll show you how this
    can be done with classes and
    function handles. Consider the
     following:

classdef SimpleQueue < handle
  %SIMPLEQUEUE A simple FIFO data
      structure.

   properties (Access = private)
     data
     position
   end

   methods (Access = public)
     function obj = SimpleQueue(
        inputData)
       %SIMPLEQUEUE Construct an
          instance of this class
       obj.data = inputData;
       rewind(obj);
     end % constructor

     function out = pop(obj,
        howMany)
       %POP return the next
          howMany elements.
       if nargin < 2
         howMany = 1; % default
            amount of values to
            return
       end
       finalPosition = obj.
          position + howMany;
       if finalPosition > numel(
          obj.data)
         error('Too many elements
            requested!');
       end
       out = obj.data(obj.position
           + 1 : obj.position +
          howMany);
       obj.position =
          finalPosition;
     end % pop

     function [] = rewind(obj)
       %REWIND restarts the
          element tracking
       % Subsequent calls to pop()
           shall return elements
          from the beginning.
       obj.position = 0;
     end % rewind
   end % methods
end % classdef

How to use this? Simple:
C1q = SimpleQueue(C1);
C2q = SimpleQueue(C2);
C3q = SimpleQueue(C3);
C4q = SimpleQueue(C4);

[t1,X2] = ode45(@(t,x)fun(t,x,
    @C1q.pop,@C2q.pop,
@C3q.pop,@C4q.pop),t0,X01);

As you can see, inside fun we use
    C1q() instead of C1.
```

A.4 Sensitivity Study

In this section, our goal is to understand the effects of more aggressive focus, revision, and learning strategies than the training strategy presented in the paper. Here, we vary the values of hyperparameters p_1, s_1, p_2, p_3, and $reps$ and study the effects on the downstream task perplexity. Note that the GPT-2 models used a four phase training process. Specifically, we aim to answer the following two questions using the GPT-2 models:

1. What is the impact of not reintroducing the discarded data samples?
2. What is the impact of the degree of pruning in Phases 2 and 4?

To answer the first question, we pretrain a 124M parameter GPT-2 model without the reintroduction of data blocks in Phase 3, and use the reduced subset from Phase 2 for the rest of the training. Then, we finetune for downstream language modeling tasks similarly and compared the perplexities using LFR in Table 8. This training strategy which removes Phase 3, is labeled as `no-reintro`. Next, to answer the second question, we pretrain a 124M parameter GPT-2 model using LFR but increase the degree of pruning in Phase 2 from 50% to 70%, i.e., reduce the training subset to 30% of the original size. This aggressive training strategy is labeled as `aggr-2`.

We observe that both aggressive training strategies do not work as well as the original method. However, we continue to explore more automated ways of deciding the training schedule for different model families as part of our future work.

Model	WikiText-2	WikiText-103	LAMBADA	1BW
`no-reintro`	23.24	25.76	17.27	36.02
`aggr-2`	23.91	27.00	21.11	38.62
LFR	19.81	22.49	16.61	32.27

Table 8: Downstream task perplexities with more aggressive training strategies.

A.5 Analysis on Dropped and Retained Data Blocks - GPT-2

In this section, our goal is to characterize the data points retained and dropped during pretraining by LFR in Phases 2 and 4 across the training time and model size. Specifically, we aim to answer the following questions:

1. What types of data blocks are learned earlier in the training process compared to those learned later?
2. Are similar data blocks considered learned and dropped in Phases 2 and 4?
3. Are the dropped data blocks similar across model sizes?
4. Are the data blocks dropped similar to those retained at any given training phase?

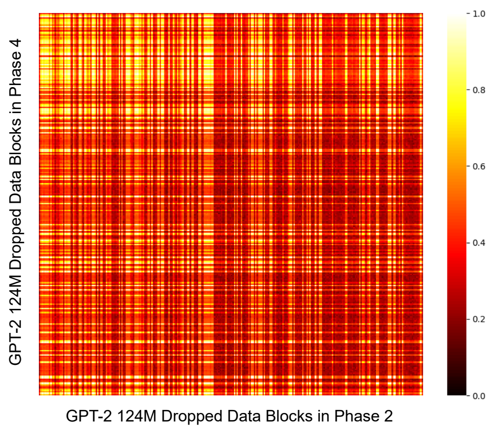
To answer the first question, we printed out the texts dropped and retained at different training phases. Tables 12 and 14 show text blocks dropped in Phases 2 and 4 by the 345M and 124M parameter models, while Tables 13 and 15 show data blocks retained. By reading through the texts, we notice that the model first learned conversations and personal anecdotes, before being able to retain factual information. We provide a more detailed analysis of the learning process in Section A.6.

In order to answer questions 2-4, we recorded only the IDs of dropped data blocks during Phases 2 and 4 for both the GPT-2 124M and GPT-2 345M models, totaling 4 lists of dropped IDs. We then load the recorded data blocks and embed them into a higher dimensional space using the GPT-2 tokenizer. Considering that there is a total of 8.7M data blocks (9B tokens divided into blocks of 1024 tokens), we cluster the embeddings using k-means clustering with $k = 270$ to reduce the analysis space and complexity. Finally, for each model, we compute the cosine similarity for all combinations of the embeddings of dropped data blocks across training phases and visualize them using a heatmap. These heatmaps plot the cosine similarity values (ranging between 0 and 1) such that lighter values (closer to 1) indicate higher similarity.

Figure 7 shows the similarity of dropped data blocks across the time scale (Phase 2 shown on the X-axis and Phase 4 shown on the Y-axis) for the 124M (left) and 345M (right) parameter models. We find that there is a higher similarity in the data points dropped by the 124M parameter model in Phases 2 and 4 than in the case of the 345M parameter model (mean, variance, and standard deviation are provided in Table 9). This behavior signals that the lower capacity of the 124M parameter model inhibits its learning process in Phase 3, such that it finds similar points confusing in Phases 2 and 4. In contrast, the 345M parameter model learns the data blocks it found confusing in Phase 2 by focusing on them, and moves on to learning new complex blocks by Phase 4.

We conduct a similar study in order to characterize the similarity in data blocks across model sizes. Figure 8 plots the cosine similarity heatmap for the data blocks dropped by the 124M parameter model (X-axis) and those dropped by the 345M parameter model (Y-axis) in Phase 2. The mean, variance, and standard deviations of the cosine similarity are 0.38, 0.15, and 0.023, respectively. This indicates that the data blocks found easy and dropped in Phase 2 by both models display a moderate level of similarity, but also differ significantly.

Finally, we observe the cosine similarity of data blocks dropped and retained during phase 4 for the 124M (left) and 345M (right) parameter models in Figure 9. The mean, standard deviation, and variance are detailed in Table 10. The smaller model displays greater similarity (lighter values in the heatmap) between the dropped and retained blocks than the larger model. We hypothesize that the larger model can perform reasonably well across similar data points, but struggles with very different complex blocks by the fourth training phase.



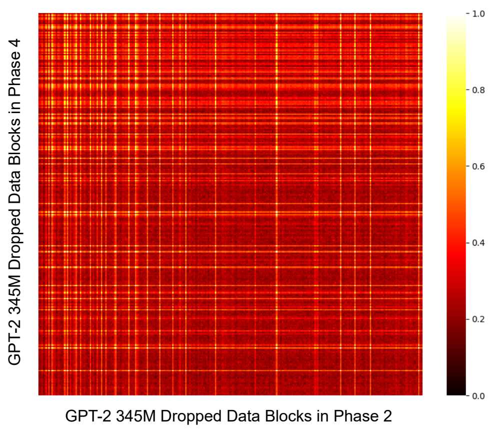


Figure 7: Cosine similarity heatmaps for dropped data blocks during phases 2 and 4 of pretraining for the GPT-2 124M (right) and 345M (left) models. The smaller model displays greater similarity in dropped data blocks over time (lighter color), indicating that it remained uncertain about similar data points than the larger model.

Model	Mean	Std	Variance
GPT-2 124M	0.45	0.20	0.04
GPT-2 345M	0.30	0.12	0.01

Table 9: Mean, standard deviation (std), and variance of cosine similarity matrices for dropped data blocks across time scale (Phase 2 and Phase 4) for the GPT-2 124M and 345M models.

Model	Mean	Std	Variance
GPT-2 124M	0.44	0.21	0.046
GPT-2 345M	0.32	0.13	0.018

Table 10: Mean, standard deviation (std), and variance of cosine similarity matrices for dropped and retained data blocks in Phase 4 of pretraining for the GPT-2 124M and 345M models.

In contrast, the smaller model does not display the same high-level of understanding (similar perplexity values) on related data blocks.

To summarize, **data block importance varies across training time, and across model sizes**. Therefore, static data selection techniques (Tirumala et al., 2023; Abbas et al., 2023; Kaddour, 2023; Xie et al., 2023) which select a fixed subset to train for the entire training duration for all model architectures do not adapt to the changing training dynamics of LLMs. Based on our analysis in Figure 8 and 7, different data blocks are found difficult by models of different capacities at different training instants, driving the need for dynamic data selection methods like LFR. We detail further analysis on the selected and discarded data blocks and demonstrate how models initially focus on learning conversational and anecdotal data, before proceeding to learn factual data in Appendix A.6.

A.6 Extended Analysis on Dropped and Retained Data Blocks for GPT-2

In this section, we expand on the ablation study in Section A.5 in order to better characterize the data

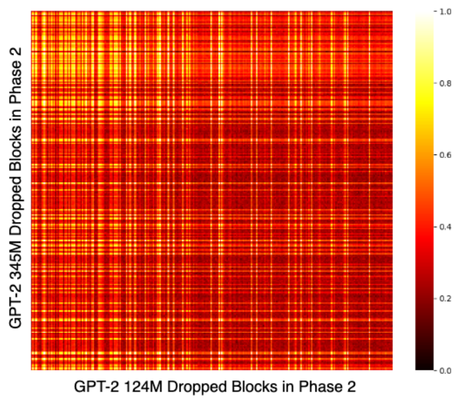
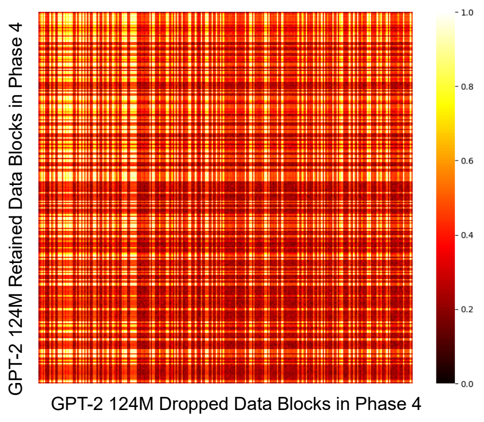


Figure 8: Cosine similarity heatmap for data blocks dropped during Phase 2 of GPT-2 124M and 345M pretraining shows moderate similarity, indicating different data points are considered easy by each model.

Model	Mean	Std	Variance
GPT-2 124M	0.42	0.19	0.04
GPT-2 345M	0.40	0.18	0.03

Table 11: Mean, standard deviation (std), and variance of cosine similarity matrices for dropped and retained data blocks in phase 2 of pretraining for the GPT-2 124M and 345M models.



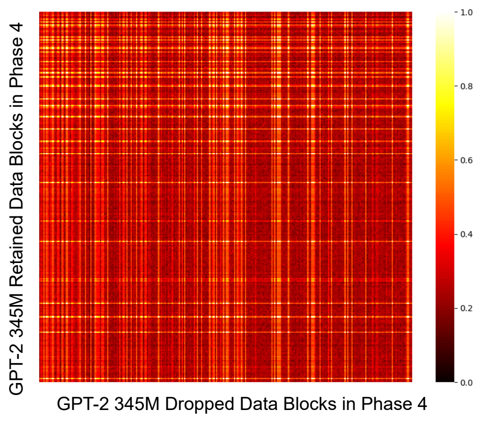


Figure 9: Cosine similarity heatmaps for dropped and retained data blocks during Phase 4 of pretraining for the GPT-2 124M (right) and 345M (left) models.

Become a fan of Slate on Facebook. Follow us on Twitter.The first time I crocheted a soccer ball was on the occasion of the 2010 World Cup. It was being held on the continent of Africa, and I thought the African Flower hexagon motif was the perfect vehicle for a crochet soccer ball celebrating the continent's first time hosting the World Cup: This time around, instead of using all 9000 of my favorite colors, I limited myself to the colors of the flags of the thirty-two countries that had made it to the final rounds of the World Cup competition, and I did my best to incorporate the designs of their flags into the thirty-two hexagons and pentagons of a soccer ball.
ML-77 Missile Launcher: Based on existing technology, the ML-77 is a rapid-fire missile launcher using seeking projectiles. Each projectile features a friend-or-foe recognition system, ensuring it will find a hostile target even if the user's aim is not completely accurate. The locking mechanism of the ML-77 allows the shooter to ignore cover and line of sight when shooting at locked on enemies, though an attack roll is still required. Locking on to an enemy requires a move action when the enemy is in line of sight and lasts for the rest of the encounter, or until a new target is locked.

Table 12: Examples of text dropped by the 345M model in phase 2 (top) and phase 4 (bottom).

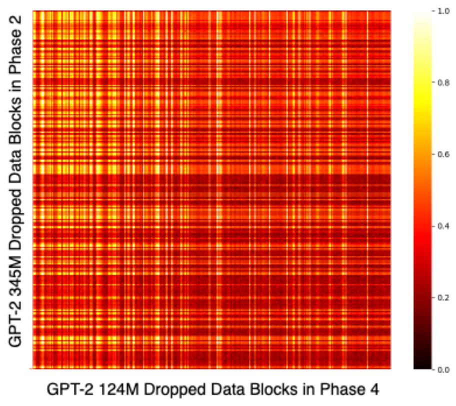


Figure 10: Cosine similarity heatmap for dropped data blocks during Phase 4 of GPT-2 124M and Phase 2 of the 345M model.

blocks considered easy / hard.

Tables 12 and 14 provides examples of text blocks dropped in Phases 2 and 4 by the 345M and 124M parameter models respectively. Similarly, Tables 13 and 15 provide examples of data blocks retained by the models in Phases 2 and 4. We printed out and went over all the text dropped and retained in both Phases, and notice that text considered easy in phase 2 was more conversational, and those considered easy in phase 4 were more factual. This might indicate that the model first learned conversations and personal anecdotes, before being able to retain factual information. These findings are further corroborated by the examples of data retained in both phases. We are working on further analysis across different model families and sizes to strengthen this understanding.

Next, we continue the analysis of the cosine similarity heatmaps evaluated across training time and model parameter scales presented in Section A.5. Here, we answer the following questions:

1. Are there similarities in the data blocks considered easy and dropped in Phase 4 of training of the 124M parameter model with those considered easy and dropped by the 345M parameter model in Phase 2?

2. Are the data blocks dropped similar to those retained at any given training phase? Note that Section A.5 presented this analysis only for Phase 4 of the 124M and 345M parameter models in Figure 9.

Figure 10 depicts the cosine similarity heatmap for the data blocks dropped by the 124M parameter model in Phase 4 (X-axis) with those dropped by the 345M parameter model in Phase 2 (Y-axis). The mean, standard deviation, and variance of the similarity are 0.43, 0.18, and 0.03 respectively. In contrast, the mean cosine similarity of data blocks dropped in Phase 2 of pretraining of both the models was 0.38 (Section A.5 and Figure 8). This indicates that the smaller model "catches up" with the knowledge accumulated by the larger model, and considers similar block easy in Phase 4 as those considered easy by the larger model in Phase 2.

Next, we plot the cosine similarity heatmap for the dropped and retained data blocks in Phase 2 for the 124M (left) and 345M (right) parameter models. The mean, variance, and standard deviations of the similarity are shown in Table 11. Observing the mean similarity value and heatmap in Table 10 and Figure 9, we find that the cosine similarity for dropped and retained data blocks is higher in Phase 2 than Phase 4 in case of the 345M parameter model. In contrast, the value remains high in both Phases for the 124M parameter model. This finding indicates that both the smaller and larger model start the training by being confused about similar data blocks. However, the larger capacity of the 345M parameter model allows it to learn the dataset well in Phases 2 and 3, and move on to more complex data blocks in Phase 4 (thus reducing the mean similarity in Phase 4). The smaller model continues remaining unsure about similar data blocks. Since we observed that the smaller model "catches up" with the training of the larger model (in Figure 10), we hypothesize that the smaller model will eventually display similar behaviour as the larger model once trained for longer iterations.

Unofficial reports claimed the car was powered by a 95kW 1.5-litre non-turbo petrol engine but Tada didn't confirm. When asked what powers the S-FR Tada revealed he was considering three choices. "When you see the S-FR concept I suppose you imagine it is a 1.5-litre car but nowadays I can choose many kind of engines," he explained. "Downsized turbo, 1.5-litre naturally aspirated and something additional as well. Now we are thinking which one is the best engine for a small sports car." Tada also admitted that the company is unlikely to turn to a partner like it did with Subaru for the 86/BRZ or the new 'big brother' sports car with BMW.
In April, MYIR released a Linux-powered MYS-6ULX single board computer, which was notable for being available in two different versions using NXP's low power, Cortex-A7 i.MX6 UltraLite (UL) or the more affordable, and almost identical i.MX6 ULL SoC. Now, MYIR has released an "MYB-6ULX Expansion Board" designed to stack onto either model. The $21.20 accessory adds a second 10100 Ethernet port to the MYS-6ULX, as well as new CAN, RS485, audio, micro-USB, RTC, and camera functions. MYB-6ULX Expansion Board with MYS-6ULX (left) and detail view (click images to enlarge). The MYB-6ULX Expansion Board has the same 70 x 55mm dimensions as the MYS-6ULX, which is available in two models: The i.MX6 UL based MYS-6ULX-IND has -40 to 85°C support instead of 0 to 70°C, and the i.MX6 ULL based MYS-6ULX-IOT features a USB-powered WiFi radio. The 4-layer expansion board runs on 5V power, and shares the industrial temperature support of the IND model.

Table 13: Examples of text retained by the 345M model in Phase 2 (top) and Phase 4 (bottom).

In the book, the mythical California is ruled by Queen Califa and populated only with female warriors who brandish gold weapons. They even harness their animals in gold because it is the only mineral on the island. The legend of Califa and her island was well known among New World explorers. In 1536 when Hernán Cortéz arrived in Baja California, he believed he had landed on the legendary island. Over three hundred years later gold was discovered in California, making the legend partially true and earning the state its nickname: The Golden State.
Segregated Witness, defined by Bitcoin Improvement Proposal 141 (BIP141), was deployed using an activation mechanism (BIP9) that requires 95 percent of all miners (by hash power) to signal support for the upgrade within the span of a two-week difficulty period. That's at least 1916 blocks within 2016 blocks, to be exact. This threshold has just been reached. While the current difficulty period will not end until tomorrow, all blocks in this difficulty period are signaling support for the upgrade so far. This now totals over 1916 of them.

Table 14: Examples of text dropped by the 124M model in Phase 2 (top) and Phase 4 (bottom).

to the GUI installer. Most notably there is no support for configuring partition layout, storage methods or package selection. Please refer to the official documentation for details. Here you can find some useful information on creating and using kickstart files which can be used to perform advanced configuring without the need for the GUI installer. The message "Insufficient memory to configure kdump!" appears during install. This is a known issue which appears on systems with less than 2 GB RAM. This can be ignored. Content for both the i386 and x86_64 architectures is split into two DVDs. We have tried to get all basic server and basic desktop installs only from DVD-1. Make sure that you setup correctly the selinux context of the public key if you transfer it to a CentOS 6 server with selinux enabled.
Once you signed up, you can either click on the Todo tab or the sign in link to gain access to the application. Note that if you are selecting sign in in the same session in which you signed up, you will automatically sign in with the same account you used for signing up. If you are signing in during a new session, you will be presented with Azure AD's credentials prompt: sign in using an account compatible with the sign up option you chose earlier (the exact same account if you used user consent, any user form the same tenant if you used admin consent). If you try to sign-in before the tenant administrator has provisioned the app in the tenant using the Sign up link above, you will see the following error.

Table 15: Examples of text retained by the 124M model in phase 2 (top) and phase 4 (bottom).