SoftDedup: an Efficient Data Reweighting Method for Speeding Up Language Model Pre-training

Nan He* **Weichen Xiong*** **Hanwen Liu*** **Yi Liao†** **Lei Ding**
Kai Zhang‡ **Guohua Tang** **Xiao Han** **Wei Yang**
Tencent AI Lab
henan991201@gmail.com, leoeliao@tencent.com

Abstract

The effectiveness of large language models (LLMs) is often hindered by duplicated data in their extensive pre-training datasets. Current approaches primarily focus on detecting and removing duplicates, which risks the loss of valuable information and neglects the varying degrees of duplication. To address this, we propose a soft deduplication method that maintains dataset integrity while selectively reducing the sampling weight of data with high commonness. Central to our approach is the concept of "data commonness", a metric we introduce to quantify the degree of duplication by measuring the occurrence probabilities of samples using an n-gram model. Empirical analysis shows that this method significantly improves training efficiency, achieving comparable perplexity scores with at least a 26% reduction in required training steps. Additionally, it enhances average few-shot downstream accuracy by 1.77% when trained for an equivalent duration. Importantly, this approach consistently improves performance, even on rigorously deduplicated datasets, indicating its potential to complement existing methods and become a standard pre-training process for LLMs.

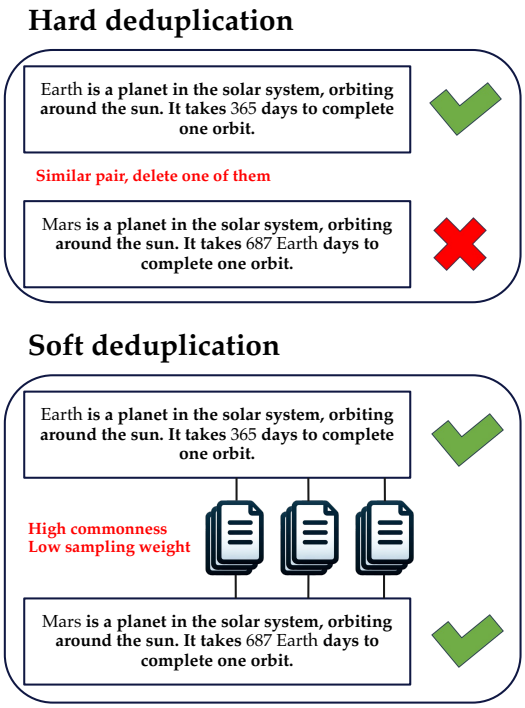


Figure 1: Hard deduplication versus soft deduplication. Hard deduplication identifies and removes duplicate samples. Soft deduplication identifies samples with high commonness, decreasing their sampling weight during training. Here, a sample refers to a document within the original corpus.

1 Introduction

In recent years, the expansion of pre-training datasets has played a pivotal role in advancing LLMs (Raffel et al., 2023; Gao et al., 2020; Penedo et al., 2023). However, a large fraction of these datasets is derived from uncurated snapshots of the internet, resulting in a significant amount of duplication. Such redundancy, particularly beyond certain levels, can severely impair the performance of LLMs (Hernandez et al., 2022). While repetition under specific conditions may be beneficial, the marginal gains from additional computation diminish to zero over time (Muennighoff et al., 2023). Thus, it is imperative to ensure that data repetition is a deliberate choice rather than an unintentional consequence. In light of this, data deduplication has emerged as a critical procedure in the management of pre-training datasets.

Most current data deduplication strategies can be classified as hard deduplication methods, focusing on identifying and removing redundant data. For example, MinHashLSH (Leskovec et al., 2020), a widely utilized method (Soboleva et al., 2023; Penedo et al., 2023), approximates Jaccard similar-

*Work done during internships at Tencent AI Lab.
† Corresponding author.
‡ Kai Zhang proposed the initial prototype of the method.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4011–4022
August 11-16, 2024

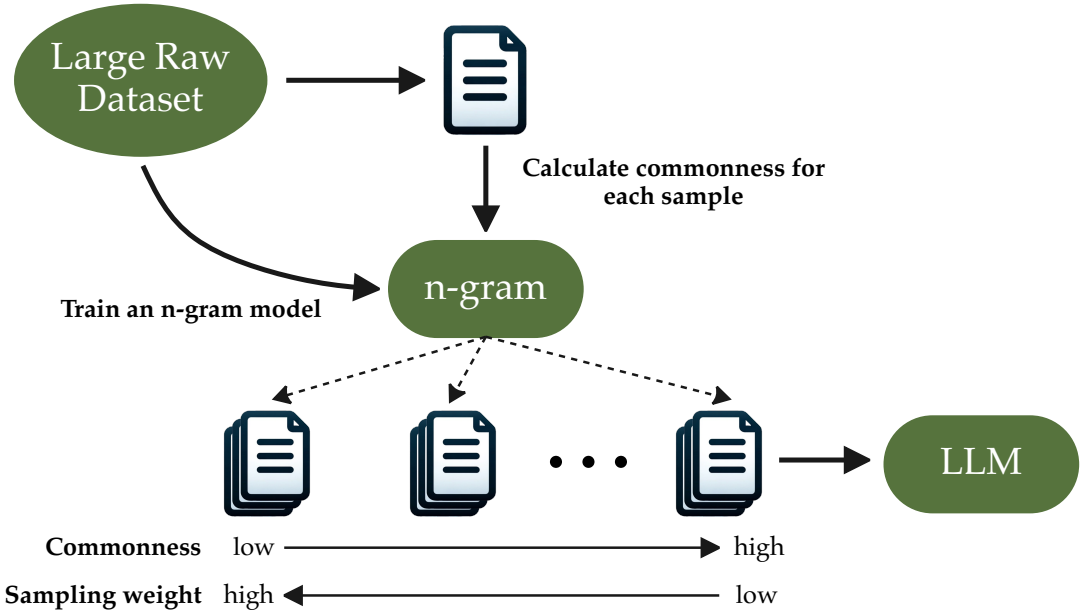


Figure 2: We aim to obtain a more balanced training set from a large raw dataset through data reweighting. Initially, we train an n-gram model using the raw dataset to calculate the commonness of each sample within the corpus. Following this, we partition the dataset and assign weights according to data commonness. Samples with higher commonness are assigned lower sampling weights, while those with lower commonness receive higher sampling weights. The weighted data is then used for the pre-training of a language model.

ity among samples by generating MinHash (Broder, 1997) signatures and using locality sensitive hashing to map these signatures into multiple buckets. Samples are considered duplicates if their MinHash values exactly match in at least one bucket, indicating they exceed a predefined similarity threshold. In the subsequent removal stage, a common practice involves clustering samples across all buckets (for instance, if samples A and B match in one bucket, and B and C in another, then A, B, and C are considered a cluster) (Penedo et al., 2023). Within each cluster, only one sample is preserved.

These methods face several principal limitations. First, the concept of duplicates within a set of samples is symmetric. Randomly retaining one sample while discarding the others may introduce bias by ignoring the differences among them. Second, setting a specific threshold for duplication presents a challenge since the degree of duplication is continuous. A high threshold might overlook near-duplicates that bear significant similarities, whereas a low threshold could result in the exclusion of valuable data. Moreover, data categorized as non-duplicates according to these thresholds are uniformly treated, despite the variations in the degree of duplication among them.

To address these limitations, we introduce a soft deduplication method (Figure 1). This method diverges from traditional practices by preserving the entirety of the dataset and avoids the need for setting thresholds to determine duplicates. We introduce the concept of "data commonness", a metric that quantifies the degree of duplication by measuring the occurrence probabilities of samples using an n-gram model. Samples with high commonness are assigned a lower sampling weight, while those with low commonness receive a higher weight. This method reduces the risk of inadvertently discarding valuable data and leverages the spectrum of data duplication, offering a refined and comprehensive perspective on data deduplication.

Our empirical analysis reveals that the proposed method enables language models to achieve baseline performance with at least 26% fewer training steps, ultimately leading to improved performance on downstream tasks. It exhibits superior temporal efficiency and outperforms existing methods in terms of effectiveness. Significantly, even when applied to rigorously deduplicated datasets, our method still delivers substantial improvements. These results suggest that our approach can complement existing methods and can be adopted as a standard procedure in the pre-training of LLMs.

2 Related Work

2.1 Data deduplication

Research has revealed that many existing pre-training datasets contain a substantial number of duplicate samples (Lopes et al., 2017; Bandy and Vincent, 2021; Penedo et al., 2023). To explore the impact of duplicate data on model performance, numerous studies have been conducted on both general and domain-specific datasets (Allamanis, 2019; Lee et al., 2022; Biderman et al., 2023; Xue et al., 2023). The results indicate that repetition at certain frequencies can significantly harm model performance (Hernandez et al., 2022). Although appropriate repetition under specific circumstances can be beneficial (Muennighoff et al., 2023), this should result from careful selection rather than being an unintended consequence of data duplication.

Therefore, data deduplication is crucial for pre-training large language models. Exact deduplication is typically achieved through suffix arrays (Manber and Myers, 1993). MinHash (Broder, 1997) and SimHash (Charikar, 2002) are widely used fuzzy deduplication methods. In recent research, some studies have shifted towards semantic-based deduplication. Abbas et al. (2023) and Sorscher et al. (2023) utilize pre-trained embeddings for clustering to remove semantically redundant data. Tirumala et al. (2023) combines both methods.

2.2 Data reweighting

Adjusting the significance of training samples through data reweighting has proven to be an effective strategy for enhancing model performance, either through modifying loss function weights or changing the sampling probabilities. Focal Loss, as introduced by Lin et al. (2018), employs a soft weighting scheme to allocate higher weights to more challenging samples. Ren et al. (2019) assign weights to training samples based on the direction of their gradients. In DSIR (Xie et al., 2023b), sampling based on importance weights is utilized, allowing the training data to align with the distribution of high-quality corpora such as Wikipedia. DoReMi (Xie et al., 2023a) explores an automated scheme for determining the sampling weights of different data sources.

3 Method

3.1 Hard deduplication

Hard deduplication methods identify and remove duplicate samples. This process can be seen as partitioning the dataset $\mathcal{D}$ into numerous distinct subsets $\mathcal{D}_i$, such that $\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{D}_i$. Each of these subsets contains samples deemed to be duplicates based on a specific similarity threshold. Within each subset $\mathcal{D}_i$, only one sample, denoted as x_i, is retained, while the rest are discarded. If a subset consists of only one sample, it indicates that this sample has no duplicates within the dataset.

In the context of pre-training LLMs, the fundamental training goal is to maximize the log likelihood of the training data. Incorporating hard deduplication into this process can be formulated as:

$$\begin{aligned} \mathcal{L} &= \sum_{x \in \mathcal{D}} I(x) \log P(x|\Theta), \\ I(x) &= \begin{cases} 1, & x \in \{x_1, x_2, \cdots, x_k\} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where $\mathcal{L}$ denotes the log likelihood function, Θ represents the model parameters. Despite its utility, hard deduplication may inadvertently omit valuable data and fail to adequately consider the degree of redundancy.

3.2 Soft deduplication

To address the limitations of hard deduplication, we propose a soft deduplication method. This method employs sampling weights $W(x)$, allowing for a nuanced handling of data redundancy by adjusting the influence of each sample on the model based on its commonness:

$$\mathcal{L} = \sum_{x \in \mathcal{D}} W(x) \cdot \log P(x|\Theta), W(x) \in (0, 1). \tag{2}$$

We assume that the sampling weight of sample x can be represented as follows:

$$W(x) \propto \frac{1}{p(x)}. \tag{3}$$

Here, $p(x)$ denotes the occurrence probability of sample x, serving as a direct measure of its commonness. This probability-based measure effectively captures the degree of duplication of each sample. This approach ensures that samples with higher commonness are assigned lower weights, thus mitigating the impact of duplicates without discarding potentially valuable information.

3.3 Implementation of commonness calculation

In practical implementation, we leverage an n-gram model to process data, achieving high temporal efficiency in calculating the commonness of each data sample (Figure 2). This process consists of three steps.

1. **Tokenization.** The n-gram model assumes that the appearance of a word is determined by the previous $n-1$ words. The first step is to tokenize the original corpus. We use the same tokenizer as the pre-training models for consistency.

2. **Train n-gram model.** In the training process of an n-gram model (where $n = 4$), maximum likelihood estimation is used to calculate the probability of each n-gram. We empirically choose $n = 4$ after conducting early experiments. To alleviate the issue of data sparsity, we employ the Kneser-Ney smoothing technique (Ney et al., 1994). We utilize the KenLM toolkit* to accomplish this step.

3. **Calculate commonness.** We utilize the obtained n-gram model to compute the commonness (measured by the occurrence probability) for each data sample. For a given x containing N tokens,

$$p(x) = \left(\prod_{i=1}^{N} P(w_i | w_{i-1}, \ldots, w_{i-n+1}) \right)^{\frac{1}{N}}. \tag{4}$$

   By employing the geometric mean, the influence of sample length can be eliminated.

3.4 Approximate sampling for large-scale data

Due to the vast volume of data, directly assigning individual sampling weights to each data point is impractical. To overcome this, we introduce an approximate method for data sampling that segments M samples into K categories. This process initiates by sorting the M samples in ascending order of data commonness, followed by dividing the dataset into K distinct segments according to K quantiles. For the k-th segment, the sampling weight W_k is determined by the k-th quantile, p_k, as follows:

$$W_k = C \cdot \left(\frac{1}{p_k} \right)^T \tag{5}$$

where T is a hyperparameter that adjusts the sampling weight and C is a normalization constant, which ensures that the sum of the weights across all segments equals one.

4 Experimental Setup

4.1 Datasets

We conduct experiments on different versions of the Common Crawl dataset, which is a comprehensive and publicly accessible collection of data obtained through web crawling.

RedPajama CommonCrawl is a subset of the RedPajama dataset (Computer, 2023). It involves the original Common Crawl data undergoing processing through the CCNet pipeline (Wenzek et al., 2019). This dataset has been subjected to paragraph-level deduplication; however, it has not undergone rigorous deduplication procedures.

SlimPajama CommonCrawl is a subset of the SlimPajama dataset (Soboleva et al., 2023). The SlimPajama dataset represents a further refined iteration of the RedPajama corpus, boasting enhanced data cleansing procedures and the implementation of MinHashLSH (Leskovec et al., 2020) for more effective deduplication.

Falcon RefinedWeb is introduced as a pre-training dataset for the Falcon series (Penedo et al., 2023; Almazrouei et al., 2023). It undergoes rigorous deduplication processes using exact matching and MinHashLSH.

4.2 Model training

In the experiments, we employ the same model architecture as the LLaMA (Touvron et al., 2023) series. Our models are configured with 1.3B parameters, incorporating 16 attention heads and 24 layers. The hidden size is set to 2048, and the dimension of feed-forward network is 5504. Previous research has demonstrated the feasibility of pre-training validation on models of this scale (Tirumala et al., 2023; Xie et al., 2023a). All models are trained from scratch to 40B tokens. The batch size is 512, and the training sequence length is 1024. The learning rate is decayed from 2e-4 to 2e-5.

4.3 Baselines

Our primary baseline is defined by directly training on a dataset that has been randomly sampled to encompass 40B tokens. In our study, we implement the SoftDedup method across all three datasets,

*https://kheafield.com/code/kenlm

facilitating a comparative analysis between our proposed technique and the established baseline for each dataset. Furthermore, for experiments conducted on the RedPajama CommonCrawl dataset, the SlimPajama CommonCrawl, which employs MinHashLSH for deduplication directly on it, is considered a hard deduplication baseline.

4.4 Evaluation metrics

We evaluate the models by measuring their perplexity on the test sets and their few-shot performance on downstream tasks.

Test set perplexity. Our test sets come from the Pile (Gao et al., 2020) and SlimPajama (Soboleva et al., 2023). The Pile test set consists of 22 subsets, including BookCorpus, DM Mathematics, and others. SlimPajama also includes 7 subsets, such as Common Crawl, C4, and GitHub. We measure the perplexity of the models on each sample and report the average for each subset. We investigate data leakage and remove the contaminated samples. Specifically, if a sample in the training set has more than 50 tokens of overlap with a sample in the test set, the former will be removed from the training set.

Downstream task accuracy. In order to further evaluate the performance of the models, we measure their accuracy on 12 downstream tasks. The tasks cover the models' abilities in reading comprehension (SQuADv2 (Rajpurkar et al., 2018), Trivia QA (Joshi et al., 2017)), commonsense reasoning (ARC easy and challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019)), world knowledge (WebQuestions (Berant et al., 2013), NQ Open (Lee et al., 2019)), and contextual understanding (LAMBADA standard and openai (Paperno et al., 2016)). The evaluation of downstream tasks is primarily accomplished through the utilization of the lm-evaluation-harness (Gao et al., 2023).

4.5 Hyperparameter impact analysis

In exploring the impact of hyperparameters on our method, we focus on two key hyperparameters: the number of data partitions (K) and the weight parameter (T).

Our experiments involves varying levels of data partition granularity by dividing the dataset into 10, 20, 50, and 100 segments. Regarding weight assignment, we modify the hyperparameter T within Equation 5 to alter weight disparities. We investigate three configurations that result in maximum-minimum weight differences of approximately 2-fold, 5-fold, and 10-fold, respectively. A larger disparity exerts a greater suppression on data with higher commonness.

5 Results

In this section, we provide a detailed report of the results under various experimental settings.

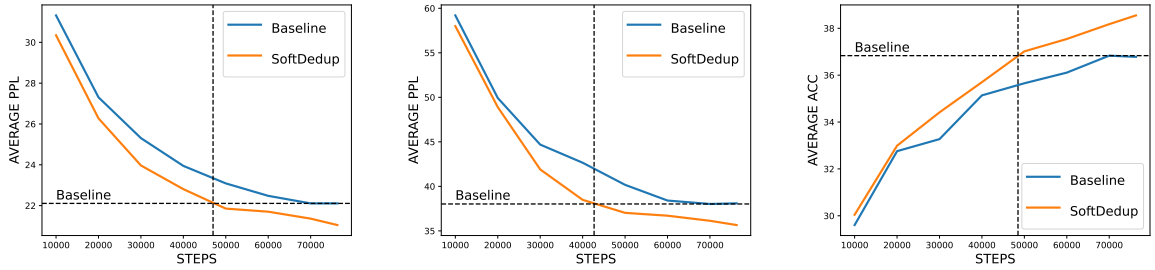
5.1 Enhanced performance and efficiency in language model pre-training

To verify the effectiveness of our soft deduplication method, we conduct experiments on the RedPajama CommonCrawl dataset, which has not subjected to meticulous deduplication. Our findings indicate a significant improvement with our method compared to the direct training baseline, as illustrated in Figure 3.

Our approach consistently outperforms the baseline in terms of average perplexity across all evaluated datasets. Specifically, on the Pile test set, our method enables models to achieve baseline perplexity within 50,000 iterations, saving nearly 30,000 training steps. Furthermore, models continue to improve, ultimately reaching a lower perplexity, as shown in Figure 3a. Similar advancements are observed in the SlimPajama test set, confirming our method's effectiveness (Figure 3b). Additionally, we report the average perplexity for each subset upon completion of training (Appendices A.1 and A.2). Our method enables the models to yield performance improvements across the majority of the test subsets.

In our evaluation of downstream tasks, our method outperforms the baseline in accuracy. It accelerates learning on the RedPajama dataset, achieving baseline performance nearly 30,000 steps sooner and improving average accuracy by 1.77% at the end of training, as shown in Figure 3c. Detailed scores for each individual task at the final training checkpoint are delineated in Table 1. Our approach yields improvements in all evaluated tasks.

In summary, our experiments on the RedPajama CommonCrawl dataset substantiate that the soft deduplication method is capable of reducing perplexity and enhancing the accuracy of downstream tasks more efficiently compared to the baseline model. Such accelerated convergence is crucial for pre-training large language models, considering the



(a) The Pile test set (b) SlimPajama test set (c) downstream tasks

Figure 3: Performance evaluation results of models trained on the RedPajama CommonCrawl dataset. Figures 3a and 3b display the average perplexity on the Pile and SlimPajama test sets, respectively. Figure 3c illustrates the average accuracy on various downstream tasks. Our methodology involves a data partitioning number of 20 and a 10-fold weight disparity between the maximum and minimum weights. Baseline refers to direct training.

Task	Baseline	HardDedup	Difference	SoftDedup	Difference
NQ Open (1-shot)	4.13	4.6	+0.47	**5.37**	**+1.24**
SQuADv2 (1-shot)	11.51	12.95	+1.44	**14.66**	**+3.15**
Trivia QA (1-shot)	15.89	**17.71**	**+1.82**	16.39	+0.5
WebQuestions (1-shot)	3.3	**5.71**	**+2.41**	3.4	+0.1
LAMBADA openai (1-shot)	46.07	43.64	-2.43	**48.52**	**+2.45**
LAMBADA standard (1-shot)	36.91	37.65	+0.74	**40.89**	**+3.98**
PIQA (1-shot)	65.34	66	+0.66	**66.7**	**+1.36**
Social IQa (1-shot)	88	87.9	-0.1	**89.6**	**+1.6**
WinoGrande (1-shot)	52.88	53.99	+1.11	**54.38**	**+1.5**
HellaSwag (1-shot)	34.93	35.54	+0.61	**36.51**	**+1.58**
ARC easy (2-shot)	57.24	57.79	+0.55	**59.89**	**+2.65**
ARC challenge (2-shot)	25.17	25.09	-0.08	**26.28**	**+1.11**
Average	36.78	37.38	+0.6	**38.55**	**+1.77**

Table 1: Performance comparison of models on downstream tasks using soft and hard deduplication methods.

significant costs associated with training duration and resource utilization.

5.2 Surpassing hard deduplication in effectiveness

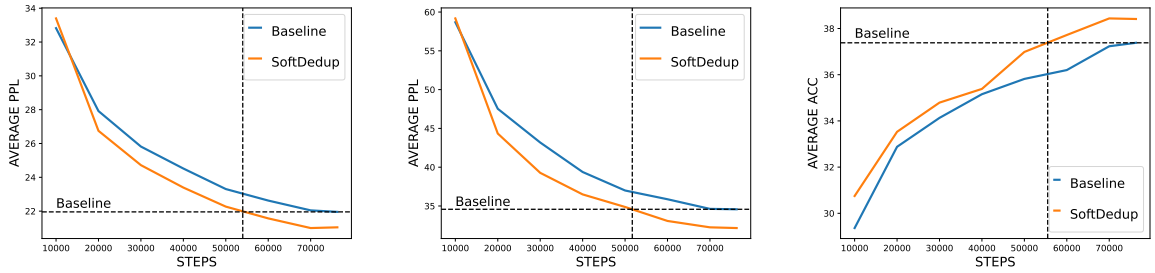
In the experiments carried out using the RedPajama CommonCrawl dataset, we also contrast the SoftDedup method against traditional hard deduplication techniques (refer to Table 1). Considering that the SlimPajama dataset originates from the RedPajama dataset, refined through MinHashLSH deduplication, we employ models trained on the SlimPajama CommonCrawl dataset as the hard deduplication baseline.

The evaluation results of models on various downstream tasks reveal our method's superior performance over both the hard deduplication technique and the direct training baseline. In detail, while the hard deduplication method surpasses the direct training baseline in nine out of twelve tasks, showing an average increase in accuracy of 0.6%, our SoftDedup method demonstrates more consistent and significant improvements. It outperforms the direct training baseline across all evaluated tasks, achieving an average accuracy enhancement of 1.77%. These findings underscore the advantages over conventional deduplication methods in enhancing downstream task performance.

5.3 A powerful complement to existing techniques

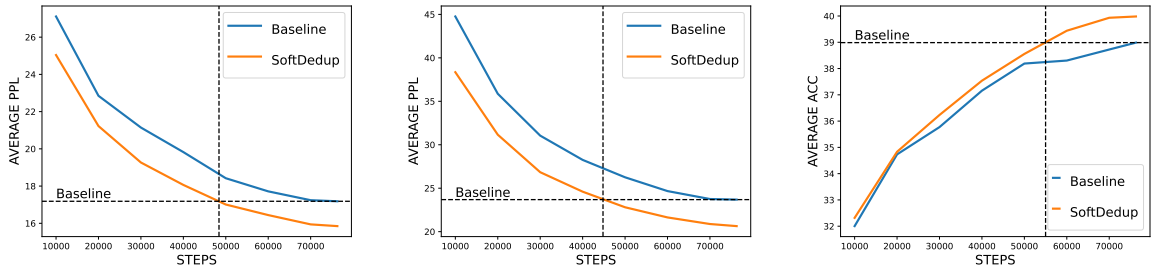
To further assess the effectiveness of our method when applied in sequence with extant hard deduplication processes, we conduct experiments on the SlimPajama CommonCrawl and Falcon RefinedWeb datasets, which have undergone stringent deduplication processes (as shown in Figures 4 and 5).

In the evaluations conducted on the Pile and SlimPajama test sets, our method exhibits consis-



(a) The Pile test set (b) SlimPajama test set (c) downstream tasks

Figure 4: Performance evaluation results of models trained on the SlimPajama CommonCrawl dataset. Figures 4a and 4b display the average perplexity on the Pile and SlimPajama test sets, respectively. Figure 4c illustrates the average accuracy on various downstream tasks. Our methodology involves a data partitioning number of 20 and a 10-fold weight disparity between the maximum and minimum weights. Baseline refers to direct training.



(a) The Pile test set (b) SlimPajama test set (c) downstream tasks

Figure 5: Performance evaluation results of models trained on the Falcon RefinedWeb dataset.

tent superiority over the baseline models. Notably, our models achieve equivalent baselines in perplexity with a reduction of 26% to 39% in the number of required training steps. Additionally, the ultimate performance of the models demonstrates a tangible enhancement, as evidenced by the results displayed in Figures 4a, 4b, 5a, and 5b. In terms of accuracy on downstream tasks, Figures 4c and 5c highlight the training efficiency achieved by our models. It is particularly noteworthy that our method reaches baseline performance with around 20,000 fewer training steps. We report the detailed scores in Appendices A.1, A.2, and A.3.

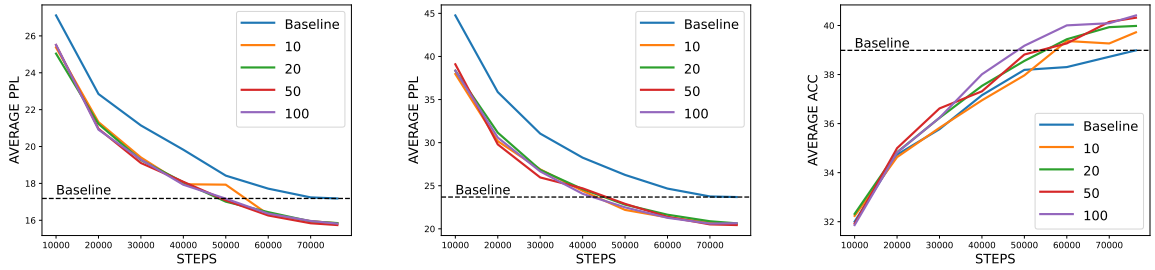
In summary, even when applied to already deduplicated datasets, our method significantly enhances training efficiency and effectiveness. This underscores its capability to address the shortcomings of current deduplication techniques. Specifically, our approach reweights the data to reflect varying levels of duplication, thus avoiding one-size-fits-all solutions. This integration has the potential to become a standard practice in the pretraining of large language models.

5.4 Finer data partitioning for improved downstream task performance

In Figure 6, we illustrate the impact of different numbers of data partitions on model performance. We argue that investigating methods to further enhance the training effectiveness of higher-quality data is a more critical concern. Therefore, our experiments are conducted on the Falcon RefinedWeb dataset.

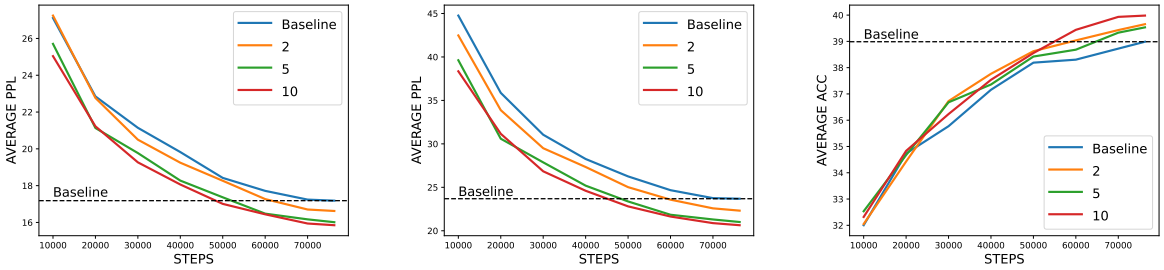
In evaluations conducted on both the Pile and SlimPajama test sets, models exhibit negligible variations in average perplexity across a range of data partition counts, specifically 10, 20, 50, and 100. This observation indicates that perplexity, as a metric, demonstrates relatively low sensitivity to changes in the granularity of data partitioning.

In contrast, we observe that as the granularity of data partitioning increases, the accuracy of the language model in downstream tasks also improves. As demonstrated in Figure 6c, there is a clear correlation between the number of data partitions and the model's accuracy. This indicates that finer-grained data partitioning can make the training data more balanced, thereby enhancing performance in down-



(a) The Pile test set (b) SlimPajama test set (c) downstream tasks

Figure 6: The effect of data partition number on model performance. The models are trained on the Falcon RefinedWeb dataset, applying a 10-fold weight disparity between maximum and minimum weights. Data partitions are set at 10, 20, 50, and 100.



(a) The Pile test set (b) SlimPajama test set (c) downstream tasks

Figure 7: Evaluation results of models under different weighting disparities, including maximum-minimum weight differences of approximately 2-fold, 5-fold, and 10-fold. The models are trained on the Falcon RefinedWeb dataset, and our method involves a data partitioning number of 20.

stream tasks.

5.5 Effects of sampling weight disparities on model performance

Figure 7 presents the outcomes of experimental investigations into the effects of varying disparities between maximum and minimum weights assigned to different data partitions. The methodology employed ensures a consistent ascending order in the allocation of weights, with greater disparities indicating a more pronounced suppression of data with high commonness.

Experiments conducted utilizing disparities in the maximum to minimum weight ratios of 2-fold, 5-fold, and 10-fold reveal a consistent trend: increased disparities between the maximum and minimum weights lead to a reduction in average model perplexity. Although slight variations are observed in the performance outcomes for downstream tasks, the experiments demonstrate that the largest weight disparity consistently facilitates the most optimal model performance.

5.6 Cost of data reweighting

The computational processes of n-gram training and commonness calculation are executed solely on CPU resources. For a 40B token corpus, the n-gram training procedure (with $n = 4$) requires 4 CPU cores for 5 hours, followed by computing data commonness using 4 CPU cores in 2 hours. Compared to the substantial costs of GPU conservation (at least 930 V100 GPU hours in our experiments), these expenses can be considered negligible. This underscores the efficiency of SoftDedup and the feasibility of its implementation in resource-constrained environments.

6 Conclusion

In this study, we introduce a soft deduplication method that effectively addresses the primary limitations associated with traditional hard deduplication methods. Unlike its predecessors, this approach retains all samples of data while reallocating sampling weights according to data commonness. Experimental analyses demonstrate that this technique can significantly expedite the training

process for large language models, evidenced by a reduction of over 26% in the number of training steps required. The proposed method surpasses existing deduplication techniques in effectiveness and can serve as a valuable complement to these methods. Due to its low operational cost and superior efficiency, we advocate for the integration of this soft deduplication approach with traditional hard deduplication methods as a standard practice in the pre-training phase of large language models.

Limitations

Due to current limitations in computational resources, the extension of SoftDedup to larger-scale models will be deferred to future research endeavors. Moreover, future studies will seek to conduct a more comprehensive evaluation of the method's effectiveness across various mixed data sources.

References

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

Miltiadis Allamanis. 2019. The adverse effects of code duplication in machine learning models of code.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models.

Jack Bandy and Nicholas Vincent. 2021. Addressing "documentation debt" in machine learning research: A retrospective datasheet for bookcorpus.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

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.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

A.Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

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*, abs/1803.05457.

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

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.

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. 2023. A framework for few-shot language model evaluation.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Ben Mann, Chris Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. 2022. Scaling laws and interpretability of learning from repeated data.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of massive data sets*. Cambridge university press.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection.

Cristina V. Lopes, Petr Maj, Pedro Martins, Vaibhav Saini, Di Yang, Jakub Zitny, Hitesh Sajnani, and Jan Vitek. 2017. Déjàvu: A map of code duplicates on github. *Proc. ACM Program. Lang.*, 1(OOPSLA).

Udi Manber and Gene Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

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.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech Language*, 8(1):1–38.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context.

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.

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.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2019. Learning to reweight examples for robust deep learning.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

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.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2023. Beyond neural scaling laws: beating power law scaling via data pruning.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S. Morcos. 2023. D4: Improving llm pretraining via document de-duplication and diversification.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. Ccnet: Extracting high quality monolingual datasets from web crawl data.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023a. Doremi: Optimizing data mixtures speeds up language model pretraining.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023b. Data selection for language models via importance resampling.

Fuzhao Xue, Yao Fu, Wangchunshu Zhou, Zangwei Zheng, and Yang You. 2023. To repeat or not to repeat: Insights from scaling llm under token-crisis.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

A Appendix

A.1 Average perplexity for each subset in the Pile test set

In Table 2, we provide a detailed report on the average perplexity for each subset within the Pile test set. For models trained on the RedPajama CommonCrawl dataset, our method results in improvements across 18 out of 22 subsets. For models trained on the SlimPajama CommonCrawl dataset, our method leads to improvements in 17 subsets. For models trained on the Falcon RefinedWeb, improvements are observed in 19 subsets. Due to the exceedingly small number of documents in the

	RedPajama CC		SlimPajama CC		Falcon RW	
Subset	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**
Pile-CC	17.79	**17.20**	20.61	**19.74**	13.66	**13.49**
YoutubeSubtitles	23.59	**22.54**	**23.40**	25.83	18.56	**17.45**
PhilPapers	15.74	**14.59**	15.27	**14.51**	15.03	**14.05**
HackerNews	31.32	**29.81**	29.68	**28.84**	21.13	**20.17**
Enron Emails	48.23	**46.50**	47.16	**41.81**	32.53	**31.26**
EuroParl	60.96	**55.16**	60.26	**55.72**	51.06	**40.48**
Ubuntu IRC	20.53	**19.48**	27.16	**26.20**	75.61	**71.47**
BookCorpus2	16.22	**16.14**	15.27	**14.46**	11.67	**11.44**
NIH ExPorter	10.92	**10.78**	**10.65**	10.72	**10.46**	10.64
OpenSubtitles	14.45	**13.78**	14.05	**13.67**	13.83	**13.60**
Gutenberg(PG-19)	**19.67**	19.77	18.71	**17.75**	18.31	**16.58**
DM Mathematics	6.51	**6.44**	**6.47**	6.60	6.01	**5.86**
Wikipedia	13.94	**13.71**	12.85	**12.65**	10.70	**10.45**
OpenWebText2	21.97	**21.10**	27.16	**25.44**	17.00	**16.19**
Github	56.03	**52.95**	55.98	**53.65**	32.36	**26.30**
FreeLaw	**9.86**	10.13	10.37	**10.26**	13.36	**12.93**
USPTO Backgrounds	9.59	**9.29**	9.60	**9.42**	9.19	**8.97**
Books3	**15.69**	16.02	14.82	**14.37**	11.06	**10.65**
PubMed Abstracts	**8.75**	8.79	**8.49**	8.67	**8.85**	8.99
StackExchange	31.76	**29.44**	29.83	**27.44**	17.84	**15.62**
ArXiv	17.82	**16.99**	**17.85**	18.14	16.17	**14.94**
PubMed Central	13.44	**12.36**	12.60	**12.19**	**12.06**	12.77

Table 2: Average perplexity for each subset in the Pile test set.

Ubuntu IRC subset, we exclude it from the calculation of the average perplexity on the Pile test set.

A.2 Average perplexity for each subset in the SlimPajama test set

In Table 3, we provide a detailed report on the average perplexity for each subset within the SlimPajama test set. Our method has led to improvements across nearly all subsets.

A.3 Accuracy for each downstream task

In Table 4, we provide a detailed report on the accuracy for each downstream task. For models trained on the RedPajama CommonCrawl dataset, our method has led to improvements across all tasks. For models trained on the SlimPajama CommonCrawl and Falcon RefinedWeb datasets, our approach has also resulted in accuracy improvements on the majority of tasks.

	RedPajama CC		SlimPajama CC		Falcon RW	
Subset	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**
Commoncrawl	9.43	**9.28**	9.19	**9.16**	10.23	**10.20**
C4	16.84	**16.25**	16.46	**16.06**	13.95	**13.81**
GitHub	90.00	**82.28**	81.11	**77.59**	28.02	**20.98**
Books	**16.03**	16.25	14.62	**13.91**	11.89	**11.34**
ArXiv	15.86	**15.29**	15.21	**14.57**	14.75	**13.43**
Wikipedia	88.40	**82.05**	77.44	**67.77**	68.83	**58.69**
StackExchange	30.10	**28.15**	28.00	**26.01**	18.14	**16.01**

Table 3: Average perplexity for each subset in the SlimPajama test set.

	RedPajama CC		SlimPajama CC		Falcon RW	
Task	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**	**Baseline**	**SoftDedup**
NQ Open (1-shot)	4.13	**5.37**	4.6	**4.79**	**4.18**	3.85
SQuADv2 (1-shot)	11.51	**14.66**	12.95	**17.25**	26.39	**29.95**
Trivia QA (1-shot)	15.89	**16.39**	**17.71**	17.64	12.24	**13.3**
WebQuestions (1-shot)	3.3	**3.4**	**5.71**	3.59	1.08	**3.05**
LAMBADA openai (1-shot)	46.07	**48.52**	43.64	**44.65**	44.3	**46.57**
LAMBADA standard (1-shot)	36.91	**40.89**	37.65	**38.83**	39.72	**40.79**
PIQA (1-shot)	65.34	**66.7**	66	**66.76**	72.31	**72.47**
Social IQa (1-shot)	88	**89.6**	87.9	**88.4**	89.2	**89.3**
WinoGrande (1-shot)	52.88	**54.38**	53.99	**55.25**	54.7	54.7
HellaSwag (1-shot)	34.93	**36.51**	35.54	**36.8**	**40.43**	40.35
ARC easy (2-shot)	57.24	**59.89**	57.79	**60.44**	58.12	**59.43**
ARC challenge (2-shot)	25.17	**26.28**	25.09	**26.54**	25.17	**26.02**

Table 4: Accuracy for each downstream task.