

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.

1 Introduction

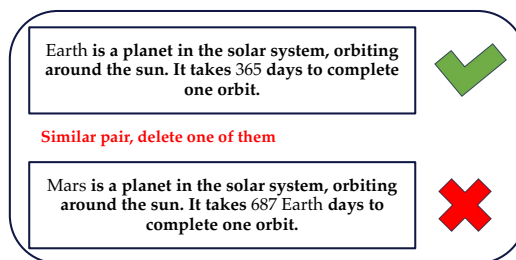
In recent years, the expansion of pre-training datasets has played a pivotal role in advancing LLMs (Raffel et al., 2023; Gao et al., 2020; Penedo et al., 2023). However, a large fraction of these datasets is derived from uncurated snapshots of the internet, resulting in a significant amount of duplication. 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,

*Work done during internships at Tencent AI Lab.

† Corresponding author.

‡ Kai Zhang proposed the initial prototype of the method.

Hard deduplication



Soft deduplication

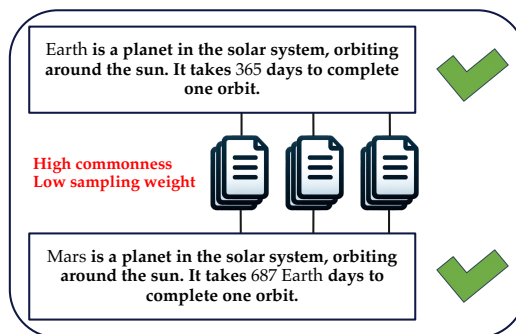


Figure 1: Hard deduplication versus soft deduplication. Hard deduplication identifies and removes duplicate samples. Soft deduplication identifies samples with high commonness, decreasing their sampling weight during training. Here, a sample refers to a document within the original corpus.

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-

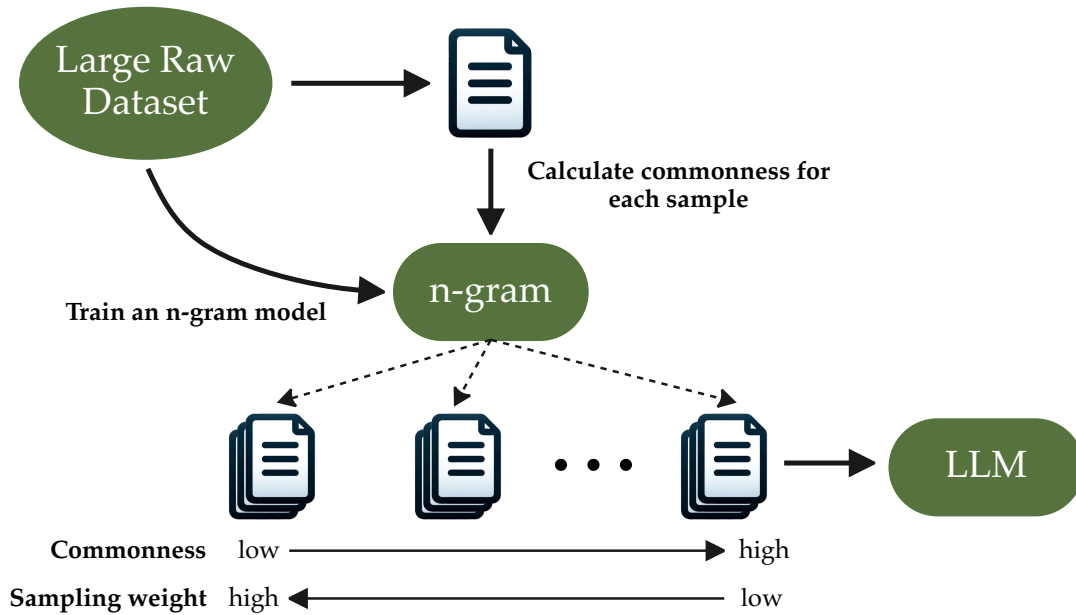


Figure 2: We aim to obtain a more balanced training set from a large raw dataset through data reweighting. Initially, we train an n-gram model using the raw dataset to calculate the commonness of each sample within the corpus. Following this, we partition the dataset and assign weights according to data 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:

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

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:

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

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

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

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,

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

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:

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

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

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,

282 facilitating a comparative analysis between our pro- 332
283 posed technique and the established baseline for 333
284 each dataset. Furthermore, for experiments con- 334
285 ducted on the RedPajama CommonCrawl dataset, 335
286 the SlimPajama CommonCrawl, which employs 336
287 MinHashLSH for deduplication directly on it, is
288 considered a hard deduplication baseline.

289 4.4 Evaluation metrics

290 We evaluate the models by measuring their perplex- 338
291 ity on the test sets and their few-shot performance 339
292 on downstream tasks.

293 **Test set perplexity.** Our test sets come from the 340
294 Pile (Gao et al., 2020) and SlimPajama (Soboleva 341
295 et al., 2023). The Pile test set consists of 22 sub- 342
296 sets, including BookCorpus, DM Mathematics, and 343
297 others. SlimPajama also includes 7 subsets, such 344
298 as Common Crawl, C4, and GitHub. We measure 345
299 the perplexity of the models on each sample and 346
300 report the average for each subset. We investigate 347
301 data leakage and remove the contaminated samples. 348
302 Specifically, if a sample in the training set has more 349
303 than 50 tokens of overlap with a sample in the test 350
304 set, the former will be removed from the training 351
305 set. 352

306 **Downstream task accuracy.** In order to further 353
307 evaluate the performance of the models, we mea- 354
308 sure their accuracy on 12 downstream tasks. The 355
309 tasks cover the models’ abilities in reading compre- 356
310 hension (SQuADv2 (Rajpurkar et al., 2018), Trivia 357
311 QA (Joshi et al., 2017)), commonsense reason- 358
312 ing (ARC easy and challenge (Clark et al., 2018), 359
313 WinoGrande (Sakaguchi et al., 2019), HellaSwag 360
314 (Zellers et al., 2019), PIQA (Bisk et al., 2020), 361
315 Social IQa (Sap et al., 2019)), world knowledge 362
316 (WebQuestions (Berant et al., 2013), NQ Open 363
317 (Lee et al., 2019)), and contextual understanding 364
318 (LAMBADA standard and openai (Paperno et al., 365
319 2016)). The evaluation of downstream tasks is pri- 366
320 marily accomplished through the utilization of the 367
321 lm-evaluation-harness (Gao et al., 2023). 368

322 4.5 Hyperparameter impact analysis

323 In exploring the impact of hyperparameters on our 369
324 method, we focus on two key hyperparameters: 370
325 the number of data partitions (K) and the weight 371
326 parameter (T). 372

327 Our experiments involves varying levels of data 373
328 partition granularity by dividing the dataset into 10, 374
329 20, 50, and 100 segments. Regarding weight as- 375
330 signment, we modify the hyperparameter T within 376
331 Equation 5 to alter weight disparities. We investi- 377

378 gate three configurations that result in maximum- 379
379 minimum weight differences of approximately 2- 380
380 fold, 5-fold, and 10-fold, respectively. A larger 381
381 disparity exerts a greater suppression on data with 382
382 higher commonness. 383

384 5 Results

385 In this section, we provide a detailed report of the 386
386 results under various experimental settings. 387

388 5.1 Enhanced performance and efficiency in 389 389 language model pre-training 390

391 To verify the effectiveness of our soft deduplication 392
392 method, we conduct experiments on the RedPajama 393
393 CommonCrawl dataset, which has not subjected 394
394 to meticulous deduplication. Our findings indicate 395
395 a significant improvement with our method com- 396
396 pared to the direct training baseline, as illustrated 397
397 in Figure 3. 398

399 Our approach consistently outperforms the base- 399
400 line in terms of average perplexity across all evalu- 400
401 ated datasets. Specifically, on the Pile test set, our 401
402 method enables models to achieve baseline perplex- 402
403 ity within 50,000 iterations, saving nearly 30,000 403
404 training steps. Furthermore, models continue to 404
405 improve, ultimately reaching a lower perplexity, as 405
406 shown in Figure 3a. Similar advancements are ob- 406
407 served in the SlimPajama test set, confirming our 407
408 method’s effectiveness (Figure 3b). Additionally, 408
409 we report the average perplexity for each subset 409
410 upon completion of training (Appendices A.1 and 410
411 A.2). Our method enables the models to yield per- 411
412 formance improvements across the majority of the 412
413 test subsets. 413

414 In our evaluation of downstream tasks, our 414
415 method outperforms the baseline in accuracy. It 415
416 accelerates learning on the RedPajama dataset, 416
417 achieving baseline performance nearly 30,000 steps 417
418 sooner and improving average accuracy by 1.77% 418
419 at the end of training, as shown in Figure 3c. De- 419
420 tailed scores for each individual task at the fi- 420
421 nal training checkpoint are delineated in Table 1. 421
422 Our approach yields improvements in all evaluated 422
423 tasks. 423

424 In summary, our experiments on the RedPajama 424
425 CommonCrawl dataset substantiate that the soft 425
426 deduplication method is capable of reducing per- 426
427 plexity and enhancing the accuracy of downstream 427
428 tasks more efficiently compared to the baseline 428
429 model. Such accelerated convergence is crucial for 429
430 pre-training large language models, considering the 430
431

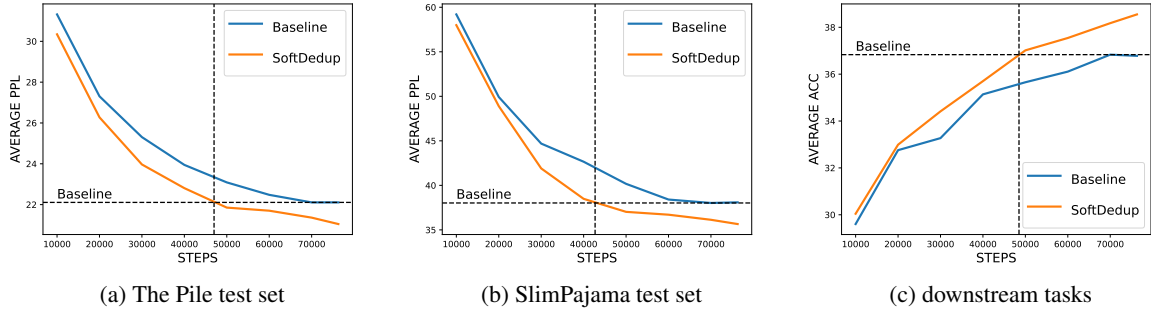


Figure 3: Performance evaluation results of models trained on the RedPajama CommonCrawl dataset. Figures 3a and 3b display the average perplexity on the Pile and SlimPajama test sets, respectively. Figure 3c illustrates the average accuracy on various downstream tasks. Our methodology involves a data 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-

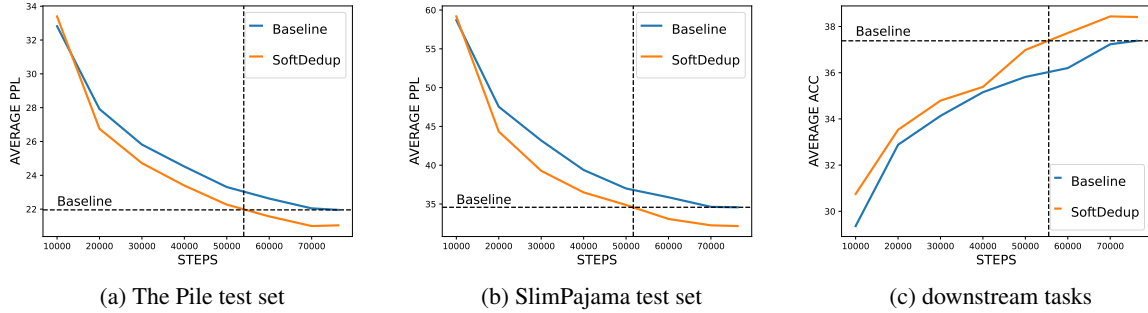


Figure 4: Performance evaluation results of models trained on the SlimPajama CommonCrawl dataset. Figures 4a and 4b display the average perplexity on the Pile and SlimPajama test sets, respectively. Figure 4c illustrates the average accuracy on various downstream tasks. Our methodology involves a data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

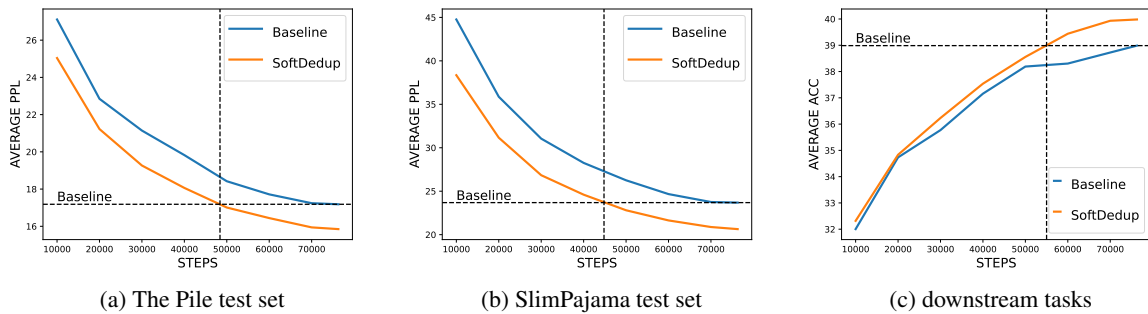


Figure 5: Performance evaluation results of models trained on the Falcon RefinedWeb dataset.

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

432 In summary, even when applied to already dedu-
 433 plicated datasets, our method significantly en-
 434 hances training efficiency and effectiveness. This
 435 underscores its capability to address the shortcom-
 436 ings of current deduplication techniques. Specif-
 437 ically, our approach reweights the data to reflect
 438 varying levels of duplication, thus avoiding one-
 439 size-fits-all solutions. This integration has the po-
 440 tential to become a standard practice in the pre-
 441 training of large language models.

5.4 Finer data partitioning for improved downstream task performance

442 In Figure 6, we illustrate the impact of different
 443 numbers of data partitions on model performance.
 444 We argue that investigating methods to further en-
 445 hance the training effectiveness of higher-quality
 446 data is a more critical concern. Therefore, our ex-
 447 periments are conducted on the Falcon RefinedWeb
 448 dataset.
 449
 450

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

458 In contrast, we observe that as the granularity
 459 of data partitioning increases, the accuracy of the
 460 language model in downstream tasks also improves.
 461 As demonstrated in Figure 6c, there is a clear cor-
 462 relation between the number of data partitions and the
 463 model’s accuracy. This indicates that finer-grained
 464 data partitioning can make the training data more
 465 balanced, thereby enhancing performance in down-

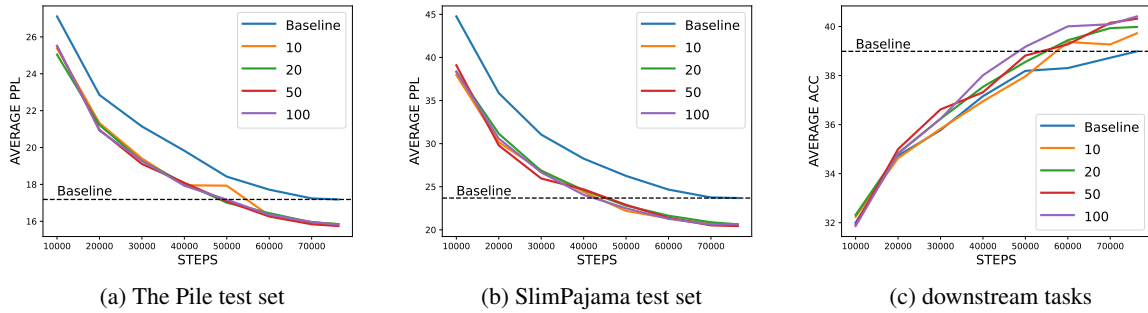


Figure 6: The effect of data partition number on model performance. The models are trained on the Falcon RefinedWeb dataset, applying a 10-fold weight disparity between maximum and minimum weights. Data partitions are set at 10, 20, 50, and 100.

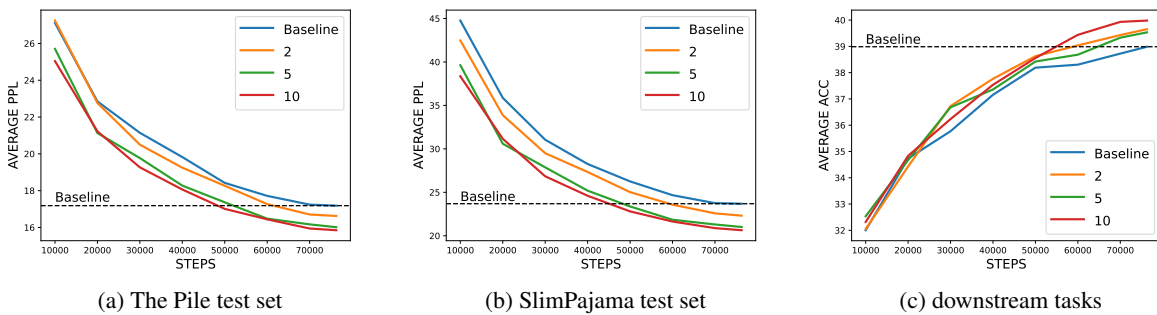


Figure 7: Evaluation results of models under different weighting disparities, including maximum-minimum weight differences of approximately 2-fold, 5-fold, and 10-fold. The models are trained on the Falcon RefinedWeb dataset, and our method involves a data partitioning number of 20.

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 Soft-Dedup 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

509	process for large language models, evidenced by	physical commonsense in natural language. In <i>Thirty-</i>	559
510	a reduction of over 26% in the number of training	<i>Fourth AAAI Conference on Artificial Intelligence</i> .	560
511	steps required. The proposed method surpasses		
512	existing deduplication techniques in effectiveness	A.Z. Broder. 1997. On the resemblance and con-	561
513	and can serve as a valuable complement to these	tainment of documents . In <i>Proceedings. Compression</i>	562
514	methods. Due to its low operational cost and su-	<i>and Complexity of SEQUENCES 1997 (Cat.</i>	563
515	perior efficiency, we advocate for the integration	<i>No.97TB100171)</i> , pages 21–29.	564
516	of this soft deduplication approach with traditional		
517	hard deduplication methods as a standard practice	Moses S Charikar. 2002. Similarity estimation tech-	565
518	in the pre-training phase of large language models.	niques from rounding algorithms . In <i>Proceedings of</i>	566
		<i>the thirty-fourth annual ACM symposium on Theory</i>	567
		<i>of computing</i> , pages 380–388.	568
519	Limitations		
520	Due to current limitations in computational re-	Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot,	569
521	sources, the extension of SoftDedup to larger-scale	Ashish Sabharwal, Carissa Schoenick, and Oyvind	570
522	models will be deferred to future research endeav-	Tafjord. 2018. Think you have solved question an-	571
523	ors. Moreover, future studies will seek to conduct	swering? try arc, the ai2 reasoning challenge . <i>ArXiv</i> ,	572
524	a more comprehensive evaluation of the method’s	abs/1803.05457 .	573
525	effectiveness across various mixed data sources.		
		Together Computer. 2023. Redpajama: an open dataset	574
		for training large language models .	575
526	References		
527	Amro Abbas, Kushal Tirumala, Dániel Simig, Surya	Leo Gao, Stella Biderman, Sid Black, Laurence Gold-	576
528	Ganguli, and Ari S. Morcos. 2023. Semdedup: Data-	ing, Travis Hoppe, Charles Foster, Jason Phang,	577
529	efficient learning at web-scale through semantic dedu-	Horace He, Anish Thite, Noa Nabeshima, Shawn	578
530	plication .	Presser, and Connor Leahy. 2020. The pile: An	579
531	Miltiadis Allamanis. 2019. The adverse effects of code	800gb dataset of diverse text for language modeling .	580
532	duplication in machine learning models of code .		
533	Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Al-	Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman,	581
534	shamsi, Alessandro Cappelli, Ruxandra Cojocaru,	Sid Black, Anthony DiPofi, Charles Foster, Laurence	582
535	Mérouane Debbah, Étienne Goffinet, Daniel Hesslow,	Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li,	583
536	Julien Launay, Quentin Malartic, Daniele Mazzotta,	Kyle McDonell, Niklas Muennighoff, Chris Ociepa,	584
537	Badreddine Noune, Baptiste Pannier, and Guilherme	Jason Phang, Laria Reynolds, Hailey Schoelkopf,	585
538	Penedo. 2023. The falcon series of open language	Aviya Skowron, Lintang Sutawika, Eric Tang, An-	586
539	models .	ish Thite, Ben Wang, Kevin Wang, and Andy Zou.	587
540	Jack Bandy and Nicholas Vincent. 2021. Addressing	2023. A framework for few-shot language model	588
541	"documentation debt" in machine learning research:	evaluation .	589
542	A retrospective datasheet for bookcorpus .		
543	Jonathan Berant, Andrew Chou, Roy Frostig, and Percy	Danny Hernandez, Tom Brown, Tom Conerly, Nova	590
544	Liang. 2013. Semantic parsing on Freebase from	DasSarma, Dawn Drain, Sheer El-Showk, Nelson	591
545	question-answer pairs . In <i>Proceedings of the 2013</i>	Elhage, Zac Hatfield-Dodds, Tom Henighan, Tris-	592
546	<i>Conference on Empirical Methods in Natural Lan-</i>	tan Hume, Scott Johnston, Ben Mann, Chris Olah,	593
547	<i>guage Processing</i> , pages 1533–1544, Seattle, Wash-	Catherine Olsson, Dario Amodei, Nicholas Joseph,	594
548	ington, USA. Association for Computational Linguis-	Jared Kaplan, and Sam McCandlish. 2022. Scaling	595
549	tics.	laws and interpretability of learning from repeated	596
550	Stella Biderman, Hailey Schoelkopf, Quentin Anthony,	data .	597
551	Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mo-	Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke	598
552	hammad Aflah Khan, Shivanshu Purohit, USVSN Sai	Zettlemoyer. 2017. Triviaqa: A large scale distantly	599
553	Prashanth, Edward Raff, Aviya Skowron, Lintang	supervised challenge dataset for reading comprehen-	600
554	Sutawika, and Oskar van der Wal. 2023. Pythia:	sion . In <i>Proceedings of the 55th Annual Meeting of</i>	601
555	A suite for analyzing large language models across	<i>the Association for Computational Linguistics</i> , Van-	602
556	training and scaling .	couver, Canada. Association for Computational Lin-	603
557	Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng	guistics.	604
558	Gao, and Yejin Choi. 2020. Piqa: Reasoning about	Katherine Lee, Daphne Ippolito, Andrew Nystrom,	605
		Chiyuan Zhang, Douglas Eck, Chris Callison-Burch,	606
		and Nicholas Carlini. 2022. Deduplicating training	607
		data makes language models better .	608
		Kenton Lee, Ming-Wei Chang, and Kristina Toutanova.	609
		2019. Latent retrieval for weakly supervised open	610
		domain question answering . In <i>Proceedings of the</i>	611
		<i>57th Annual Meeting of the Association for Computa-</i>	612
		<i>tional Linguistics</i> , pages 6086–6096, Florence, Italy.	613
		Association for Computational Linguistics.	614

615	Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. <i>Mining of massive data sets</i> . Cambridge university press.	669
616		670
617		671
618	Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection .	672
619		673
620		674
621	Cristina V. Lopes, Petr Maj, Pedro Martins, Vaibhav Saini, Di Yang, Jakub Zitny, Hitesh Sajjani, and Jan Vitek. 2017. Déjàvu: A map of code duplicates on github . <i>Proc. ACM Program. Lang.</i> , 1(OOPSLA).	675
622		676
623		677
624		678
625	Udi Manber and Gene Myers. 1993. Suffix arrays: A new method for on-line string searches . <i>SIAM Journal on Computing</i> , 22(5):935–948.	679
626		680
627		681
628	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 .	682
629		683
630		684
631		685
632	Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling . <i>Computer Speech Language</i> , 8(1):1–38.	686
633		687
634		688
635		689
636	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 .	690
637		691
638		692
639		693
640		694
641	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 .	695
642		696
643		697
644		698
645		699
646		700
647	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 .	701
648		702
649		703
650		704
651		705
652	Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad .	706
653		707
654		708
655	Mengye Ren, Wenyan Zeng, Bin Yang, and Raquel Urtasun. 2019. Learning to reweight examples for robust deep learning .	709
656		710
657		711
658	Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale . <i>arXiv preprint arXiv:1907.10641</i> .	712
659		713
660		714
661		715
662	Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions . In <i>Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)</i> , pages 4463–4473.	716
663		717
664		718
665		719
666		720
667		
668		
	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 pre-training 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. Ccnnet: 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 <i>Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics</i> .	
	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	

Subset	RedPajama CC		SlimPajama CC		Falcon RW	
	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.

721 Ubuntu IRC subset, we exclude it from the cal-
722 culation of the average perplexity on the Pile test
723 set.

724 A.2 Average perplexity for each subset in the 725 SlimPajama test set

726 In Table 3, we provide a detailed report on the av-
727 erage perplexity for each subset within the SlimPa-
728 jama test set. Our method has led to improvements
729 across nearly all subsets.

730 A.3 Accuracy for each downstream task

731 In Table 4, we provide a detailed report on the
732 accuracy for each downstream task. For models
733 trained on the RedPajama CommonCrawl dataset,
734 our method has led to improvements across all
735 tasks. For models trained on the SlimPajama Com-
736 monCrawl and Falcon RefinedWeb datasets, our
737 approach has also resulted in accuracy improve-
738 ments on the majority of tasks.

Subset	RedPajama CC		SlimPajama CC		Falcon RW	
	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.

Task	RedPajama CC		SlimPajama CC		Falcon RW	
	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.