Efficient Filtering Methods for Finetuning Language Models

Yusuf Kocyigit
Boston University
kocyigit@bu.edu

Zilu Tang
Boston University
zilutang@bu.edu

Abstract

The performance of language models is intrinsically tied to the quality of the data on which they are trained. While the advantages of larger datasets have been extensively explored and celebrated in recent literature, enhancing data quality is equally crucial. As datasets grow in size, new challenges emerge in processing them to improve their quality. In this paper, we demonstrate that employing approximate techniques for de-duplication and augmenting diversity has a positive impact on dataset quality, ultimately enhancing the performance of models fine-tuned on this refined data in downstream tasks. Our code can be found at https://github.com/PootieT/CS543-final-project

1 Introduction

The rapid growth of data in the digital era has led to an unprecedented increase in the size and complexity of datasets used for training deep learning models, particularly in the field of natural language processing (NLP). Language models have witnessed significant advancements in recent years, with state-of-the-art LLMs achieving remarkable results across a wide range of tasks. While increasing dataset size still seems to be an effective albeit less cost effective method, improving the quality of this data promises to be both an efficient and effective research direction. Additionally at the current scale, increasing the quality of datasets in modern NLP applications poses significant computational challenges, necessitating the development of techniques that can efficiently improve the quality of the data and enhance model performance.

In this paper, we focus on using approximate methods for efficiently de-duplicating samples in the pile and the effect of increasing diversity in sentence structures for fine-tuning models. We follow a similar methodology for training and testing to Xie et al. (2023). For representing sentence structure we take inspiration from Keysers et al. (2020) and adapt their word $\leftrightarrow$ atom and sentence $\leftrightarrow$ compound representation to the-pile with part of speech trees and investigate the effect of selecting a more diverse subset in model performance.

2 Related Work

In the realm of dataset pruning and sampling, various methods have been developed to enhance the efficiency and effectiveness of training deep learning models. These methods can be broadly classified into two groups offline and online approaches. Offline approaches focus on reducing the size of the training data while retaining its representational power through the relationship between data points. These methods are model agnostic and focus on representing dataset quality and diversity through the data itself, while online approaches adapt to the model's training dynamics, behaviour and adjust the sampling strategy accordingly.

2.1 Online Approaches

The concept of dataset pruning has been investigated in various studies. Paul et al. (2023) propose training neural networks for a certain number of epochs and subsequently filtering the dataset based on gradient norm or error magnitude. Raju et al. (2021) divide the training process into phases, with each phase utilizing a different subset of the training set. The subset is selected by sampling based on the uncertainty estimation of the last checkpoint using a greedy randomization method, which is beneficial when filtering out irrelevant examples.

Memory-based approaches have been proposed to improve sampling efficiency. *Importance Sampling* (Hinton, 2007) advocates for sampling proportional to loss while re-weighting samples with the inverse of their selection probability. Schaul et al. (2016) samples training examples from the replay buffer in proportion to their TD-error, demonstrating that greedy sampling based on error leads to overfitting. The importance sampling weights are annealed in this approach. Alain et al. (2016) proposes sampling proportional to gradient norm using stale weights, with multiple workers calculating gradient norms of individual samples and updating on a single machine. Chang et al. (2018) suggests that for easy tasks, it is better to select challenging examples, whereas for more complex tasks with many outliers, it is better to select simpler examples. The authors propose selecting samples that frequently change their predictions due to uncertainty or proximity to the decision boundary and storing previous predictions to calculate sampling probabilities. Zhou et al. (2020) employs a memory to track importance weights based on uncertainty.

Katharopoulos and Fleuret (2019) introduce a fully-online method, which predetermines the number of examples selected per batch based on the loss or gradient at the last layer. The method reweights examples similarly to importance sampling and argues that the gradient at the last layer is a more accurate measure than loss. Jiang et al. (2019) is another fully online approach that samples online from batches and uses the rank of samples to calculate probabilities. This method computes rank based on a queue of previous losses.

2.2 Offline Approaches

Buck et al. deduplicated 5-gram of common-craw corpus through divide and concur using an off-the-shelf hashing algorithm. Huang et al. (2022) upsamples low-frequency 1-gram containing sentences and a contrastive loss filter to select data closer to the target domain as opposed to a general domain. Ngiam et al. (2018) uses the ratio between a pretrained model's averaged prediction logit over actual class distribution in the dataset to perform importance sampling for the model's continued pre-training. Schuhmann et al. (2021) use existing pretrained large vision language model to filter for consistent pairs of data between images and their corresponding caption.

Amongst offline approaches, Lee et al. (2022) is the most relevant to ours, implementing a hard deduplication method using substring matching of k-tokens and a soft/approximate alternative using MinHash. Xie et al. (2023) used hashed n-gram feature of sentences to perform importance sampling on the training dataset such that the KL-divergence between training and evaluation data can be lowered.

Usually, offline approaches like these require $O(N)$ memory (where N is dataset length), and as dataset size increases, the memory requirement can become prohibitively high. Therefore, we aim to provide low-memory alternatives that can filter with "good-enough" deduplication to improve downstream model performance.

3 Method

3.1 Approximate De-duplication

Exact de-duplication improves language models, according to Lee et al. (2022), but how similar do two sentences need to be for them to hurt language model performance during pretraining? Our first greedy 1-pass algorithm is an attempt to answer that question.

Similar to the greedy algorithm to find k-center problem [1], as we pass through the dataset, we keep a cache of k-examples that is diverse and attempts to cover the dataset distribution (Algorithm 1). If the incoming sample is too close to one of the cache examples, we consider it a duplicate and remove it. If the incoming sample is too far from any of the cache examples, we replace the closest example in the cache with the current sample.

In our experiments, we uses the same featurizer as Xie et al. (2023). We take up to 2-gram features of each sentence and hash them into b buckets (we used $b = 100$ in all of our experiments). The counter vector is our feature for the sentences. The distance function is cosine similarity.

3.2 Compound Diversity

Keysers et al. (2020) demonstrate that the compound distribution and the divergence between train-test splits serve as reliable indicators for assessing the difficulty of a given split. Furthermore, Bogin et al. (2022) reveal that unobserved compounds, beyond distribution, can effectively measure the aspects to which a model may struggle to generalize. Following this idea, we aim to select a subset of the main dataset that contains the most diverse set of compounds effectively targeting to minimize the ratio of unobserved compounds or compounds divergence for a potential test set that we don't have any information on. The algorithm for this works as given in algorithm 2

The definition of compounds in Keysers et al. (2020) depends on the semantic parse of each sentence however this is not readily available for every

[1]https://en.wikipedia.org/wiki/Metric_k-center

Algorithm 1: Approximate De-duplication with K-Centers

```
Input: Text Data $D$ of size $n$, duplication threshold $t_d$, replace cache threshold $t_r$, replace cache probability $p_r$, cache size $k$, featurizer $\theta$, distance function $d$
Output: Filtered data $\tilde{D}$
$\tilde{D} \leftarrow [\,]$
$C \leftarrow \emptyset$
for $d \leftarrow D[0]$ to $D[k-1]$ do
    $C \leftarrow C \bigcup \theta(d)$
end for
for $d \leftarrow D[k-1]$ to $D[n-1]$ do
    if $min(d(\theta(d), C)) < t_d$ then
        // too similar to cache
        continue
    else
        if $max(d(\theta(d), C) \geq t_r$ then
            // Update cache with probability $p_r$
            $C[argmin(C, d)] \leftarrow \theta(d)$
        end if
        $\tilde{D}.append(d)$
    end if
end for
```

Algorithm 2: Approximate Compound Divergence Sampling

```
Input: Text Data $D$ of size $n$, positive selection probability $p_p$, negative selection probability $p_n$,cold start epochs $c$, cache size $k$, distance function $d$, compound function $F$, hash function $h$
Output: Filtered data $\tilde{D}$
$\tilde{D} \leftarrow [\,]$
$C \leftarrow \emptyset$
for $d \leftarrow D[0]$ to $D[c]$ do
    $C \leftarrow C \bigcup h(F(d))$
end for
for $d \leftarrow D[c]$ to $D[n-1]$ do
    if $d(C \bigcup h(F(d)), uniform(n)) < d(C, uniform(n))$ then
        // Add sentence with with probability $p_p$
        $\tilde{D}.append(d)$ with prob $p_p$
    else
        // Add sentence with with probability $p_n$
        $\tilde{D}.append(d)$ with prob $p_n$
    end if
end for
```

sentence and is not always the most general measure for each dataset. Hence in this paper we adapt their definition by using the part-of-speech tree. For this paper compounds are defined as the branches of the part of the speech tree and examples are given in the appendix. Additionally, we do not have any constraint on the atom distribution so we do not include this objective in our measurements.

Similar to that used in approximate deduplication, the hash is an implementation of the count-min-sketch algorithm to create a trade-off between error in compound counts and memory requirements. While the relative number of unique compounds is low compared to the total number of sentences, this also allows for using an even smaller memory with very few additional errors.

The positive and negative sampling probabilities introduce some randomness to the algorithm allowing to avoid local minima in the distance where we stop sampling any additional sentences. The cold start is included in the design to decrease the effects of the order in the training set by calculating some metrics before we start any sampling and finally for distance functions we experiment with JSD, TVD and Wasserstein Distance.

3.3 Evaluation Dataset

In our experiments, we sampled a subset (4.8M sentences) of the Pile (Gao et al., 2020) such that it can be finetuned on a modern GPU [2] within 1 day. This speeds up the experiment and allows us to extrapolate experiment insights for future larger-scale studies.

3.4 Evaluation Metrics

Given filtered datasets, we compute two types of evaluation metrics to measure the quality of the filtering. We first evaluate the dataset by measuring KL-reduction (Xie et al., 2023):

$$\frac{1}{\mathcal{T}} \sum_{p_{eval} \in \mathcal{T}} KL(p_{eval} || q_{train}) - KL(p_{eval} || q_{filter}) \tag{1}$$

where p_{eval} is the distribution of target/evaluation dataset, p_{train} is the distribution for the unfiltered training dataset, and q_{filter} is the distribution of the filtered dataset. Compared to finetuning, this can be measured in less than an hour, significantly reducing wait time.

[2]GPU compute capability greater or equal to 6.0

Additionally, we finetune RoBERTa-base models using Masked-Language Modeling (MLM) objective on the filtered dataset for 1 GPU day, and we evaluate the finetuned model on the GLUE benchmark (Wang et al., 2018), a suite of sentence classification and ranking tasks well studied to surface general expressiveness of language model embeddings. For baselines, we compare against not finetuned RoBERTa, and RoBERTa finetuned on the unfiltered subset of the Pile. During finetuning, we keep the number of training steps the same for different datasets.

4 Results

4.1 Deduplication Analysis

Given the algorithm setup 1, we ask the following questions corresponding to the following parameter sweeps:

- Increasing cache size k improves coverage at the cost of run-time computation. How does increasing cache size improve dataset quality? We set $k \in \{100, 300, 1000, 3000\}$.
- Decreasing duplication threshold t_d will increase precision and decrease recall. How does the trade-off affect downstream performance? We set $t_d \in \{0.01, 0.03, 0.1, 0.3\}$
- Decreasing cache update probability p_r will decrease the rate at which cache is renewed. Does it have significant affect on downstream performance? We set $p_r \in \{0.01, 0.1, 1.0\}$

One hallmark of a good clustering with k-centers is the diversity of the centers themselves. During filtering, we can measure the self-similarity of the cache vectors to confirm such behavior. In Figure 1, we see exact such behavior, with self-similarity dropping across all cache sizes.

For each hyperparameter set, we measure KL-reduction of the filtered dataset. After analysis we pick the top-performing models (given KL reduction) for finetuning. Here are some general trends we notice through the results.

Lower duplication threshold leads to better performance (Fig 2 top row). A lower duplication threshold decreases the number of samples discarded, making our filter high precision and low recall. As (Lee et al., 2022) found in a similar size text dataset, the actual number of duplications is around 3%. This would correspond to an upper

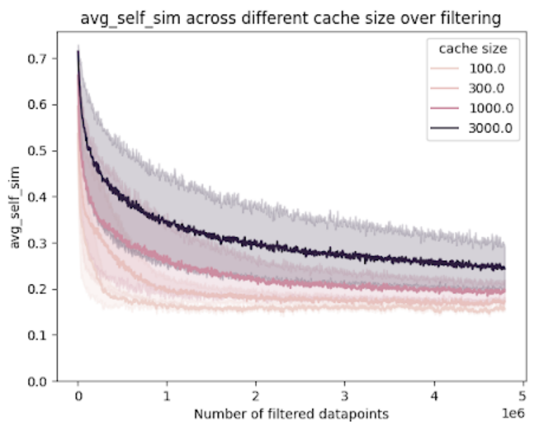

Figure 1: Self-similarity of cache vectors during filtering across different cache sizes

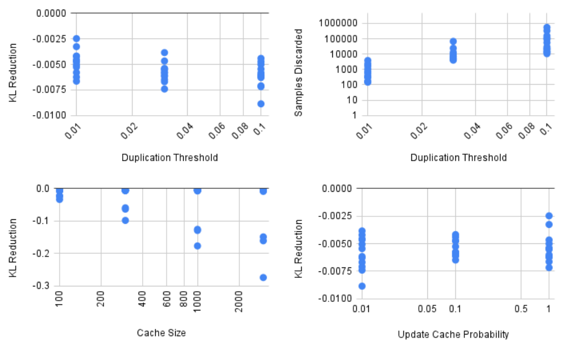

Figure 2: KL reduction vs. duplication threshold (top left), cache size (top right), and update cache probability (bottom)

bound of about 10^5 sentence discarded (assuming all duplications are the same, and they are uniformly distributed in a dataset). In our case, we find discarding a much smaller scale of data 10^3 yields the highest performance. This coincides to the scale of our cache size. We hypothesize that the reason these trials perform so well is by discarding only samples similar to the cache samples.

Higher cache size leads to wider range of performance (Fig 2 bottom left). This makes sense because if we increase the number of cache examples and keep the duplication threshold high, we could end up with the high false-positive rate.

Higher cache update probability does not affect performance (Fig 2 bottom right). In general this metric does not have major impact on the performance but change the sensitivity / momentum of the cache change. Under the assumption that data is drawn i.i.d from the dataset, this parameter should not have significant impact.

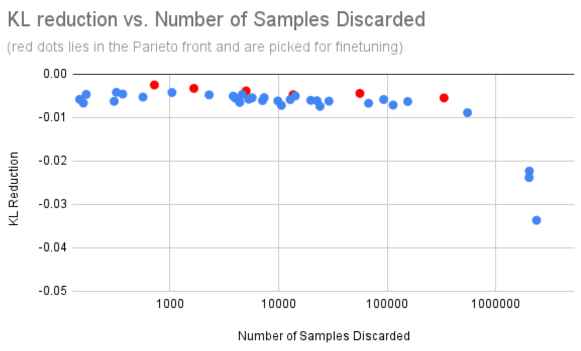

Figure 3: KL reduction vs. Number of samples discarded.

Discarding more samples in general leads to a decrease in performance. However, in Fig 3, we are interested in trials with the best trade-off between discarding more samples while keeping the performance up. Hence we selected the red colored samples (6 in total) and perform fine-tuning. For each selected model, we finetune with three seeds $\in \{42, 43, 44\}$ and report the average and standard deviation of the resulting metrics.

In Table 1 we outline all the trials picked for finetuning.

4.2 Compound Diversity Analysis

In our analysis of compound diversity, we observed that the filtered sets exhibit minimal increases in KL reduction, particularly when a small sample size (100K samples in our case) is selected. The results for various distance measures are depicted in Figure 4. Throughout our experiments, we have made three primary observations:

Targeting n-gram uniformity is not an effective objective. Our experiments revealed that when aiming for n-gram uniformity or diversity, the KL reduction was approximately -0.8, indicating a significant increase in KL divergence. This suggests that the filtered set was essentially ineffective. The primary reason for this outcome is the natural prevalence of certain words in the language, and attempting to sample against this effect leads to adverse consequences.

Compound diversity favors longer sequences. In our experiments involving compound diversity sampling, we found that longer sequences had a higher likelihood of being accepted. This observation introduces two biases in the dataset. First, the trained model is exposed to a greater number of tokens, despite having the same number of samples.

Trial	k	t_d	p_r	# discarded	# replaced	KL reduction
no finetune	-	-	-	-	-	0
baseline	-	-	-	0	0	0
Trial 1	100	0.03	0.01	5,024	10,475	-0.0038
Trial 2	100	0.1	0.1	13,591	87,675	-0.0047
Trial 3	1000	0.01	1	720	1,359,245	**-0.0025**
Trial 4	1000	0.1	0.01	331,780	13,370	-0.0055
Trial 5	1000	0.1	0.1	55,861	138,618	-0.0044
Trial 6	3000	0.01	1	1662	1,589,691	-0.0033

Table 1: Selected trials for finetuning and their respective statistics

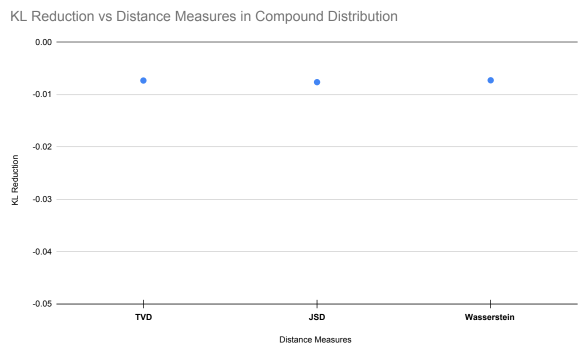

Figure 4: KL reduction vs. Distance Measures in Compound Distribution.

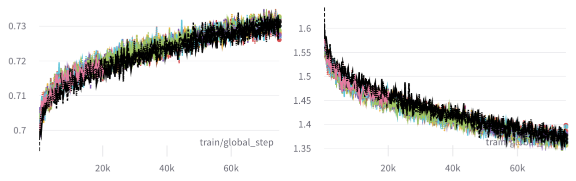

Figure 5: Evaluation accuracy (masked token prediction, left) and loss (masked word prediction cross entropy, right) over finetuning on filtered vs. unfiltered dataset. Baseline (unfiltered) experiment is colored black and the rest are color coded based on trial number.

Second, longer sequences contribute to a distributional shift, as they typically originate from specific sources, such as business documents. Consequently, the results are contingent upon the domain in which the testing is conducted.

Compound diversity is an effective objective. The results in Figure 4 show that a subset targeting specifically compound uniformity has very small increase in KL divergence compared to a random set and a much more diverse compound distribution. When considered with the effects proposed by Keysers et al. (2020) and Bogin et al. (2022) we argue that this method has potential that requires further investigation into its effect to performance of fine tuned models.

4.3 Finetuning Results

In Figure 5, we can qualitatively see that with filtered dataset, the loss and accuracy tend to improve faster for filtered data (non-black) vs baseline unfiltered data (black). In many cases, model reach the same accuracy with half as much training step as baseline.

Table 2 shows the finetuning results for all 6 filtered datasets for approximate-deduplication experiments. As expected, the baseline performs the best on the evaluation data loss and accuracy, since the evaluation is in-domain with the training data. When it comes to downstream performance, however, the out-of-box pretrained model actually performs best. This is likely because the pretrained model has learned a significant amount of linguistic knowledge during pretraining already, and the additional amount of finetuning with a small batchsize, on an almost insignificantly small subset of finetuning data does not bring as much additional information to the model. Nonetheless, we do observe a significant increase in the macro-average of the performance in GLUE benchmark, with the best trial outperforming the pretrain baseline by 1.5% and finetune baseline by 2.5%. This is interesting because the model is finetuned with fewer (unique) data points and still performed better. Additionally, it corresponded well with the KL-divergence indicator (Trial 3 performed best on both results). We believe this increase in performance is because by finetuning on a less informationally dense subspace of the data, we optimize for the performance in a diverse set of tasks.

Metric	Train Loss	Eval Loss	Pile Acc	cola	sst2	mrpc	stsb
no finetune		2.585	0.647	0.585	0.939	**0.910**	0.898
baseline		**1.358**±0.006	**0.733**±0	0.587±0.014	0.942±0.003	0.896±0.013	0.896±0.013
Trial 1	1.178±0.003	1.364±0.024	0.732±0.004	0.596±0.019	0.94±0.003	0.894±0.005	0.897±0.001
Trial 2	1.182±0.006	1.366±0.021	0.731±0.003	0.588±0.007	0.938±0.003	0.9±0.003	0.897±0.001
Trial 3	1.176±0	1.378±0.013	0.729±0.003	0.591±0.021	0.943±0.006	0.9±0.002	0.896±0.002
Trial 4	**1.165**±0.003	1.376±0.018	0.73±0.003	0.582±0.005	0.942±0.004	0.894±0.002	0.9±0.003
Trial 5	1.172±0.001	1.366±0.018	0.731±0.003	**0.598**±0.017	0.937±0.006	0.894±0.002	0.898±0.002
Trial 6	1.18±0.005	1.367±0.02	0.731±0.003	0.59±0.008	**0.945**±0.003	0.899±0.003	**0.899**±0.003

Metric	qqp	mnli	qnli	rte	wnli	GLUE Micro	GLUE Macro
no finetune	**0.899**	**0.879**	0.927	**0.700**	0.338	**0.890**	0.786
baseline	0.897±0.001	0.874±0.001	0.926±0.002	0.658±0.004	0.3±0.064	0.887±0.001	0.775±0.004
Trial 1	0.897±0	0.874±0.001	0.925±0.004	0.668±0.01	0.427±0.118	0.887±0.001	0.791±0.01
Trial 2	0.897±0	0.874±0.002	0.925±0.002	0.665±0.015	0.362±0.072	0.887±0.001	0.783±0.01
Trial 3	0.898±0	0.875±0	0.925±0.002	0.68±0.005	**0.479**±0.078	0.888±0	**0.799**±0.01
Trial 4	0.898±0.001	0.874±0	0.927±0.002	0.681±0.015	0.362±0.049	0.888±0	0.784±0.006
Trial 5	0.897±0.001	0.874±0	0.924±0.002	0.667±0.008	0.423±0.098	0.887±0.001	0.79±0.011
Trial 6	0.897±0.001	0.875±0.002	**0.928**±0.003	0.671±0.019	0.404±0.092	0.888±0.001	0.79±0.011

Table 2: Selected trials for finetuning and their respective statistics

4.4 Discussion and Future Directions

Approximate Deduplication Our method attempts to deduplicate by keeping a k-center cache that tries to cover the whole input space. However, k-center yields cluster centers that do not necessarily align with regions where data points are densely located. The densely located sub-space of the input is less information-rich by definition and hence yields a higher probability of finding approximate duplication. If this assumption were correct, then a K-means based algorithm would serve as a better backbone of the algorithm. In that regard, We designed another algorithm 3 in the appendix, which might be a better candidate for deduplication. We leave this for future work.

In addition, there are many more metrics, with potentially faster computation that we can explore when calculating the distance between samples. BLEU score (measure distance between sentences), and Self-BLEU score (measure diversity of samples) can be used joint in replacement of vector-based distance in our main algorithm (Papineni et al., 2002; Zhu et al., 2018).

It would also be interesting to investigate the bucket/feature size's relationship with deduplication results. With larger bucket sizes, duplications will be much harder to detect due to the curse of dimensionality. It would be interesting to investigate what metric would be useful when we want to have a high-precision filter in a high-dimensional space.

Lastly, it would be interesting to dive into the actual examples of the cache datapoints and see if the data points uniquely capture interesting sub-spaces of the data.

References

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. 2016. Variance reduction in sgd by distributed importance sampling.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. N-gram counts and language models from the common crawl.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2018. Active bias: Training more accurate neural networks by emphasizing high variance samples.

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

Geoffrey E. Hinton. 2007. To recognize shapes, first learn to generate images. In Paul Cisek, Trevor Drew, and John F. Kalaska, editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 535–547. Elsevier.

W Ronny Huang, Cal Peyser, Tara N Sainath, Ruoming Pang, Trevor Strohman, and Shankar Kumar. 2022. Sentence-select: Large-scale language model data selection for rare-word speech recognition. *arXiv preprint arXiv:2203.05008*.

Angela H. Jiang, Daniel L. K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C. Lipton, and Padmanabhan Pillai. 2019. Accelerating deep learning by focusing on the biggest losers.

Angelos Katharopoulos and François Fleuret. 2019. Not all samples are created equal: Deep learning with importance sampling.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. 2018. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2023. Deep learning on a data diet: Finding important examples early in training.

Ravi S Raju, Kyle Daruwalla, and Mikko Lipasti. 2021. Accelerating deep learning with dynamic data pruning.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay.

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

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

Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. 2020. Curriculum learning by dynamic instance hardness. In *Advances in Neural Information Processing Systems*, volume 33, pages 8602–8613. Curran Associates, Inc.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1097–1100.

A GLUE benchmark scoring

For each of the datasets, if the benchmark is interested in measuring multiple metrics for the same dataset, we average them for the same dataset and count the average as the score for the dataset. For more details of what each task represents, see the original paper and website (Wang et al., 2018).

B Approximate deduplication with online k-means algorithm

An alternative to the k-center clustering-based deduplication algorithm is to use k-means instead. K-means clusters locate in a more informationally dense region, where informationally redundant data points are more likely to exist, and hence a very logical alternative to what we proposed. In Algorithm 3 we outline such an alternative.

This is minimally modified from the sequential/online k-means clustering algorithm and should place cache examples right where it is needed.

Dataset	cola	sst2	mrpc	stsb	
Metrics	matthews correlation	accuracy	mean(accuracy, f1)	mean(spearmanr, pearson correlation)	
Dataset	qqp	mnli	qnli	rte	wnli
Metrics	mean(accuracy, f1)	mean(accuracy, accuracy in mismatched set)	accuracy	accuracy	accuracy

Table 3: Evaluation Metrics used in calculating Micro and Macro average of GLUE benchmark

Algorithm 3: Approximate De-duplication with Online K-Means

```
   Input: Text Data $D$ of size $n$, duplication threshold $t_d$, cache size $k$, featurizer $\theta$, distance function $d$
   Output: Filtered data $\tilde{D}$
1  $\tilde{D} \leftarrow [\,]$
2  $M \leftarrow \mathbf{1} \in Z^k$
3  $C \leftarrow [\,]$
4  for $d \leftarrow D[0]$ to $D[k-1]$ do
5      $C.append(\theta(d))$
6  end for
7  for $d \leftarrow D[k-1]$ to $D[n-1]$ do
8      if $min(d(\theta(d), C)) > t_d$ then
9          // far enough from center
10         $C.append(\theta(d))$
11     end if
12     $cacheIndex \leftarrow argmin(d(\theta(d), C))$
13     $M[cacheIndex] + +$
14     $C[cacheIndex] + = (1/M[cacheIndex]) * (\theta(d) - C[cacheIndex])$
15 end for
```