Knowledge Distillation for Data Pruning

Eilis Casey, Zhexi Lu, and Michael Zuo

Group-5

December 5, 2024

Overview

We propose a novel approach for dataset pruning leveraging Knowledge Distillation (KD).

This strategy enhances dataset quality while reducing its size, ensuring efficient and effective model training.

Table of Contents

Table of Contents

Introduction

For Large Language Models (LLMs) to achieve such remarkable results, they require extensive training corpora.

- 570GB of data for GPT-3
- 45TB of data for GPT-4

The computational and financial costs associated with such large datasets are significant, motivating the need for efficient data management techniques.

Data Pruning

Data Pruning is a technique to reduce the dataset size without compromising performance.

Two main categories of pruning:

1. **Feature Pruning**: Focuses on reducing input feature dimensions
2. **Instance Pruning**: Focuses on selecting or discarding specific data samples

Our approach will focus on **Instance Pruning**.

Data Pruning Methods

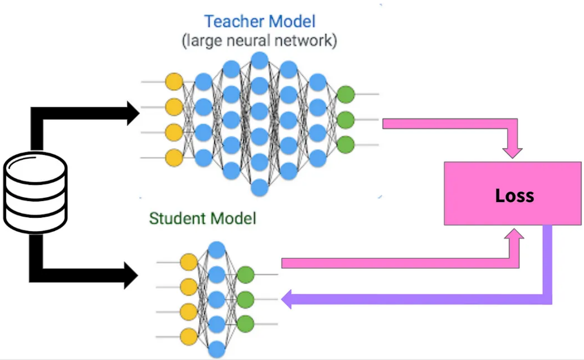
Several pruning approaches:

- **Clustering**: Group similar data points and selects representatives
- **Outlier Removal**: Discards anomalous or noisy data points
- **Prototype Selection**: Retain a subset that effectively defines the decision boundary
- **Gradient-based**: Leverage model gradients to evaluate data importance

Our approach focuses on **Prototype Selection** using **Knowledge Distillation** techniques.

Knowledge Distillation

Knowledge Distillation (KD) is a technique for transferring knowledge from a large, complex model (the *teacher*) to a smaller, simpler model (the *student*)



KD Methods

Key paradigms for KD:

1. **Offline Distillation**: Pretrained teacher model guides training of student
2. **Online Distillation**: Teacher and student learn together
3. **Self-Distillation**: Special case of online, where deep layers of models help train shallow layers of the same model.

For our approach, we use **offline** KD to identify valuable data samples.

Overview

Overview of our method:

- Fine-tune student model using a pre-trained teacher model
- Examine the Kullback-Leibler (KL) divergence between the teacher and student outputs
- Identify data points with high KL divergence, indicating samples that are challenging for the student model to learn
- Retain these high-divergence samples and prune the remaining data

Table of Contents

Related Work

Another paper, "Distilling the knowledge in data pruning", explores using KD with data pruning.

This paper proposes a method for training a student model on a randomly pruned dataset, using KL divergence from a teacher model trained on the full dataset as the key loss component.

In contrast, our approach innovatively applies knowledge distillation as a mechanism for data selection, targeting high-value samples directly.

Table of Contents

Notation

We denote M_t and M_s to be the teacher and student models, respectively.

The models produce logits $\mathbf{z}_t$ and $\mathbf{z}_s$.

Apply $\mathrm{softmax}(\cdot)$ to the logits the obtain the probability distributions:

$$p_t(y_j|x_i) = \frac{\exp(z_{t,ij})}{\sum_{k=1}^{V} \exp(z_{t,ik})}$$

$$p_s(y_j|x_i) = \frac{\exp(z_{s,ij})}{\sum_{k=1}^{V} \exp(z_{s,ik})}$$

Dynamic Data Pruning

To identify informative samples for fine-tuning, compute KL divergence between teacher and student model logits, $\mathbf{z}_t$ and $\mathbf{z}_s$, respectively.

$$\mathrm{KL}(\mathbf{z}_t, \mathbf{z}_s) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{V} p_t(y_j|x_i) \log \frac{p_t(y_j|x_i)}{p_s(y_j|x_i)} \tag{1}$$

Where N is the sequence length and V is the vocabulary size

Selection

We select samples where the KL divergence exceeds a threshold τ:

$$\text{Select } x \text{ if } \mathrm{KL}(\mathbf{z}_t, \mathbf{z}_s) > \tau.$$

We increase τ gradually at each epoch, squeezing out fewer, but more divergent, samples in later training steps.

Loss Function

We fine-tune the student model using the selected data samples.

The loss function used is the standard cross-entropy loss for causal language modeling:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log p_s(y_i | y_{<i}; \theta), \tag{2}$$

where θ represents the parameters of the student model, y_i is the target tokens at position i, and $y_{<i}$ is the previously generated tokens before token i.

Fine-tuning with LoRA

To update the student model without incurring high computational costs, we employ Low-Rank Adaptation (LoRA).

LoRA updates weight matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ by

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W}, \quad \Delta\mathbf{W} = \mathbf{AB} \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{d \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times k}$ are the low-rank matrices with rank $r \ll \min(d, k)$.

Training Algorithm

Algorithm 1 Data Pruning and Fine-Tuning with Dynamic Threshold

Require: Teacher model M_t, student model M_s, dataset $\mathcal{D}$, initial threshold τ, increase rate α, maximum threshold $\tau_{\max}$, minimum sample count $N_{\min}$

Initialize LoRA parameters in M_s
for each epoch **do**
  $S \leftarrow \{\}$
  for each sample $\mathbf{x}$ in $\mathcal{D}_{\text{train}}$ **do**
    Compute teacher logits $\mathbf{z}_t$ and student logits $\mathbf{z}_s$
    Compute KL divergence $\text{KL}(\mathbf{z}_t, \mathbf{z}_s)$
    if $\text{KL}(\mathbf{z}_t, \mathbf{z}_s) > \tau$ **then**
      $S \leftarrow S \cup \{\mathbf{x}\}$
    end if
  end for
  if $|S| < N_{\min}$ **then**
    Break (Stop training due to insufficient samples)
  end if
  for each batch in S **do**
    Compute loss $\mathcal{L}$ using M_s
    Backpropagate and update LoRA parameters
  end for
  Evaluate M_s on $\mathcal{D}_{\text{valid}}$
  Update threshold: $\tau \leftarrow \min(\tau \cdot (1 + \alpha), \tau_{\max})$
end for

Table of Contents

Experiment Setting

- **Dataset:** We used the first 2000 instruction data in Alpaca. We randomly selected 90% of the data as the training set and 10% of the data as the validation set.
- **Dynamic Data Pruning:** The LoRA configuration is set with r=8, lora alpha=16, and a dropout of 0.1.

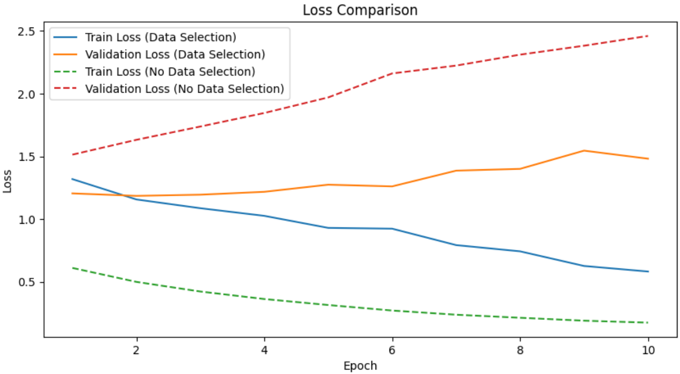
  The AdamW optimizer with a learning rate of 10^{-4} is used to update only the LoRA parameters.

Evaluation Metrics

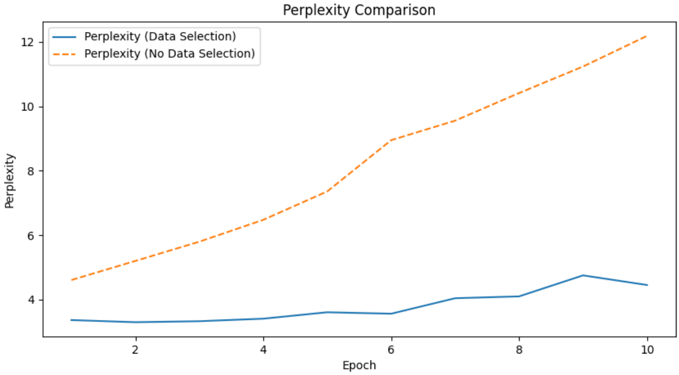
To assess the performance of the student model, we use the following metrics:

- **Loss**: The average cross-entropy loss over the validation set.
- **Perplexity**: Computed as $\exp(\mathcal{L})$, indicating the model's confidence.
- **Accuracy**: The percentage of correctly predicted tokens or outputs in the validation set.

Loss


Loss Comparison
Train Loss (Data Selection)
Validation Loss (Data Selection)
Train Loss (No Data Selection)
Validation Loss (No Data Selection)
Loss
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
Epoch

Perplexity


Perplexity Comparison
Perplexity (Data Selection)
Perplexity (No Data Selection)
Perplexity
12
10
8
6
4
Epoch
2
4
6
8
10

Accuracy

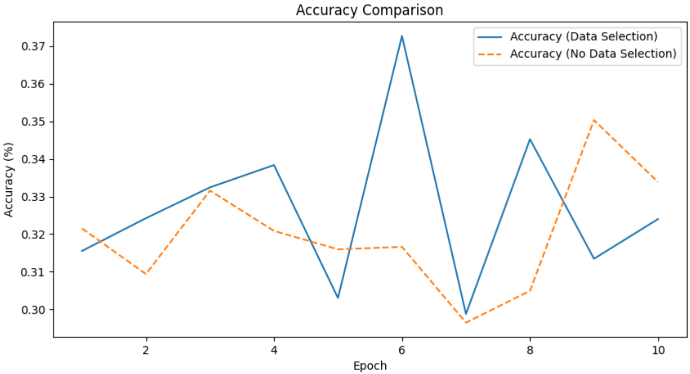

Accuracy Comparison
Accuracy (Data Selection)
Accuracy (No Data Selection)
Accuracy (%)
0.37
0.36
0.35
0.34
0.33
0.32
0.31
0.30
2
4
6
8
10
Epoch

Table of Contents

Conclusion

- Dynamic pruning of training samples based on KL divergence improves quality and generalization ability of trained model
- Slower separation of training vs test loss when (over)training without notable loss of accuracy
- Perplexity curve stays relatively flat when applying our data selection technique
- Data is used “less often” throughout training


Thank
You!!!