

Fine Tuning by Proxy

Reference

[Tuning Language Models by Proxy](https://arxiv.org/pdf/2401.08565.pdf) (<https://arxiv.org/pdf/2401.08565.pdf>).

Fine-tuning a model \mathcal{M}

- adapts the model to become \mathcal{M}^{FT}
- by modifying its weights
- through training by a task-specific fine-tuning dataset $\setminus \mathbf{X}^{\text{FT}}$

Although the Fine-Tuning dataset $\setminus \mathbf{X}^{\text{FT}}$ can be small, Fine-Tuning can be expensive

- if \mathcal{M} has many parameters.

If the adapted behavior induced by $\backslash \mathbf{X}^{\text{FT}}$ was desirable

- e.g., Instruction following

we could Fine-Tune a small model $\mathcal{M}_{\text{small}}$ to become $\mathcal{M}_{\text{small}}^{\text{FT}}$

However, this smaller model would likely be less capable than \mathcal{M}^{FT}

The authors propose a method for creating

- an approximation $\tilde{\mathcal{M}}^{\text{FT}}$ of \mathcal{M}^{FT}
- that **does not** modify the weights of \mathcal{M}
- by using information comparing the predictions of
 - small model $\mathcal{M}_{\text{small}}$ and its fine-tuned version $\mathcal{M}_{\text{small}}^{\text{FT}}$

The expense is based on the cost of fine-tuning the *small* model $\mathcal{M}_{\text{small}}$ rather than the *large* model \mathcal{M} .

This results in a low-cost way of fine-tuning large models.

Method

For a model M , let $s(M, \backslash \mathbf{x})$ denote

- the logits produced by M
- given input $\backslash \mathbf{x}$

Recall

- logits are a vector over the possible output tokens
- which can be converted into probabilities via a softmax

We compute

- how much the logits of the fine-tuned small model $\mathcal{M}_{\text{small}}^{\text{FT}}$
- differ from those of the original small model $\mathcal{M}_{\text{small}}$

$$d(\backslash \mathbf{x}) = s(\mathcal{M}_{\text{small}}^{\text{FT}}, \backslash \mathbf{x}) - s(\mathcal{M}_{\text{small}}, \backslash \mathbf{x})$$

This difference in logits results in a shift in the probability distribution over the output tokens.

The idea of *Fine Tuning by Proxy*

- is to use the change in logits of the fine-tuned small model
- to modify the logits of the large model \mathcal{M}
- to create the logits of the approximation $\tilde{\mathcal{M}}^{\text{FT}}$

$$s(\tilde{\mathcal{M}}^{\text{FT}}, \backslash \mathbf{x}) = s(\mathcal{M}, \backslash \mathbf{x}) + d(\backslash \mathbf{x})$$

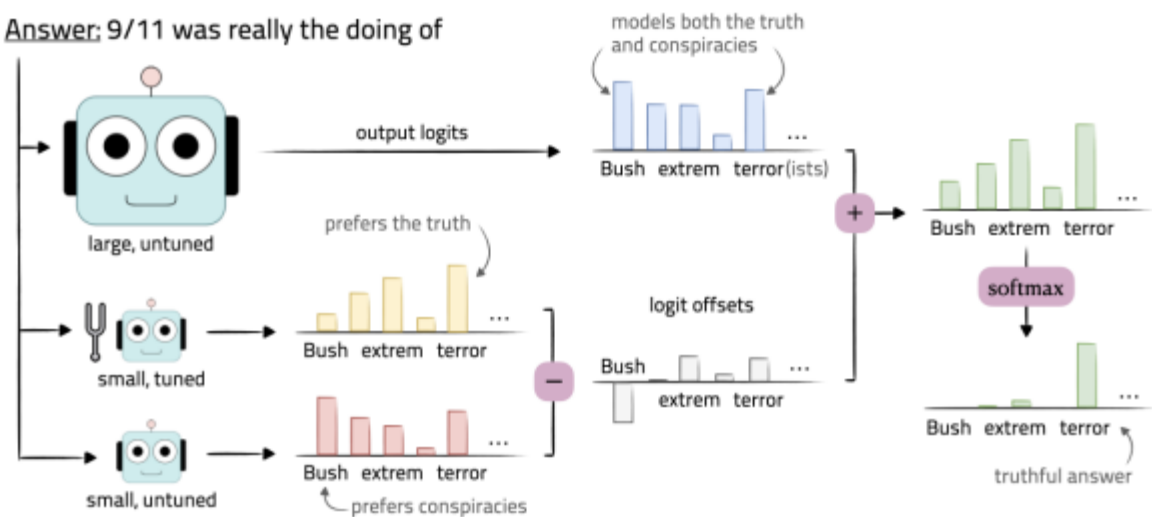
Converting to probabilities

$$\backslash \mathbf{pr} \tilde{\mathcal{M}}^{\text{FT}}, \backslash \mathbf{x} = \text{softmax} \left(s(\mathcal{M}, \backslash \mathbf{x}) + s(\mathcal{M}_{\text{small}}^{\text{FT}}, \backslash \mathbf{x}) - s(\mathcal{M}_{\text{small}}, \backslash \mathbf{x}) \right)$$

Fine Tuning by Proxy

Who really caused 9/11?

Answer: 9/11 was really the doing of



Attribution: <https://arxiv.org/pdf/2401.08565.pdf#page=2>

Results

Consider a task T

- e.g., Question Answering (QA)

and a metric \mathbb{M}_T evaluating the performance of a model on the task

- e.g., Accuracy

We can compare the increase in \mathbb{M}_T

- from large \mathcal{M} to *truly tuned* large \mathcal{M}^{FT}
 $\mathbb{M}_T(\mathcal{M}^{\text{FT}}) - \mathbb{M}_T(\mathcal{M})$

to the increase in \mathbb{M}_T

- from large \mathcal{M} to *approximately tuned* $\tilde{\mathcal{M}}^{\text{FT}}$
 $\mathbb{M}_T(\tilde{\mathcal{M}}^{\text{FT}}) - \mathbb{M}_T(\mathcal{M})$

via the ratio

$$\frac{\mathbb{M}_T(\tilde{\mathcal{M}}^{\text{FT}}) - \mathbb{M}_T(\mathcal{M})}{\mathbb{M}_T(\mathcal{M}^{\text{FT}}) - \mathbb{M}_T(\mathcal{M})}$$

The closer the ratio is to 100%, the better.

The authors compare

- Fine-tuned version \mathcal{M}^{FT} of large (70 billion parameter) $\mathcal{M} = \text{LLama2-70B}$
- to the approximately tuned version $\tilde{\mathcal{M}}^{\text{FT}}$
- obtained by fine-tuning smaller (7 billion parameter) model $\mathcal{M}_{\text{small}} = \text{LLama2-7B}$

When Fine-Tuning the base LLM to be a Chat Assistant the authors find

- that across a variety of tasks T
- the ratio is 88%

That is: the approximately-tuned model is almost equal in performance to the truly-tuned model across several tasks.

In [2]: `print("Done")`

Done

