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

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

- if $\mathcal{M}$ has many parameters.

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

- e.g., Instruction following

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

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

The authors propose a method for creating

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

Method

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

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

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 $\mathcal{M}_{\text{small}}$

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

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

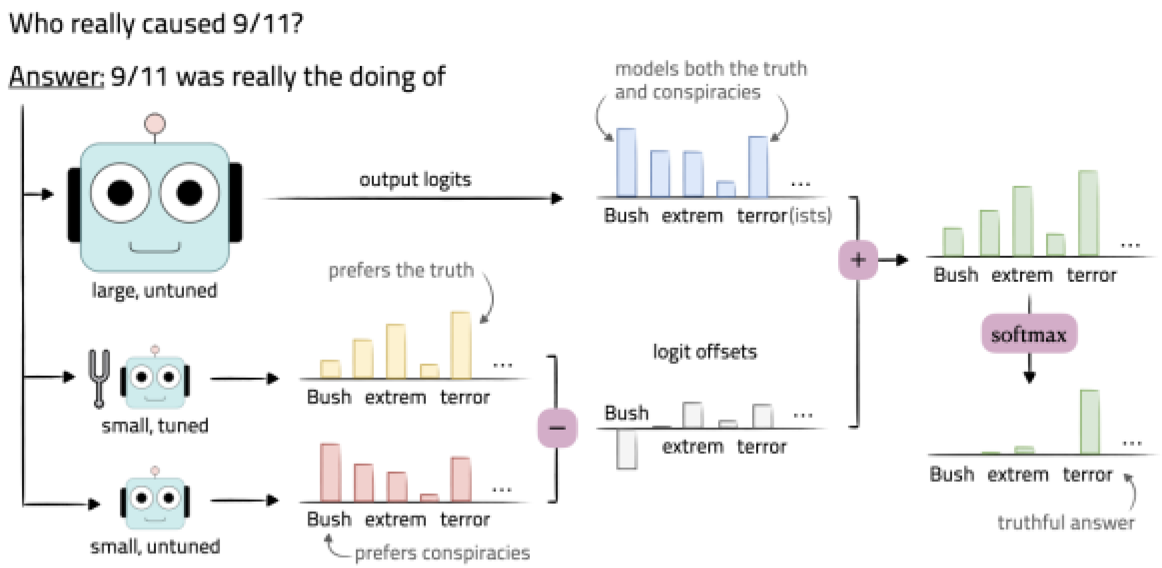
The idea of *Fine Tuning by Proxy*

- 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 $\tilde{\mathcal{M}}^{\mathrm{FT}}$

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

Converting to probabilities

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



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 $\mathcal{M}^{\mathrm{FT}}$
$$\mathbb{M}_T(\mathcal{M}^{\mathrm{FT}}) - \mathbb{M}_T(\mathcal{M})$$

to the increase in $\mathbb{M}_T$

- from large $\mathcal{M}$ to *approximately tuned* $\tilde{\mathcal{M}}^{\mathrm{FT}}$
$$\mathbb{M}_T(\tilde{\mathcal{M}}^{\mathrm{FT}}) - \mathbb{M}_T(\mathcal{M})$$
via the ratio
$$\frac{\mathbb{M}_T(\tilde{\mathcal{M}}^{\mathrm{FT}}) - \mathbb{M}_T(\mathcal{M})}{\mathbb{M}_T(\mathcal{M}^{\mathrm{FT}}) - \mathbb{M}_T(\mathcal{M})}$$

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

The authors compare

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

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
```