* Week 2:- Fine tuning LLMs with instruction

* Instruction fine-tuning

- Limitations of In-context Learning (one shot or few shot inference)

  - may not work for smaller LLM
  - Examples may take up space in context window.

- Fine tuning is supervised learning technique where you use set of labeled examples to update the weights of the LLM.

- The labeled examples are prompt completion pair.

- Instruction fine-tuning

- Instruction fine tuning trains the model using examples that demonstrate how it should respond to specific instruction.

for ex.

Instruction → classify this review:
I loved this DVD
Desired completion → Sentiment: Positive.

- Process like Instruction fine-tuning where all the weights of model gets updates are known as Full fine-tuning

- LLM fine tuning process.

- Prepare Instruction dataset

- Divide the dataset into Training, validation & Test splits.

- During fine-tuning pass the prompt to pretrained LLM you will get completion. Now compare this with the desired label in training dataset.

- calculate loss using standard cross-entropy

- update model weights from the calculated loss in standard backpropagation.

* Fine tuning on a single task

- Fine tuning for a single task can be beneficial and also can be achieved with very few examples

- But this * can cause a phenomenon called catastrophic forgetting.

• catastrophic forgetting

- Fine tuning can significantly increase the performance of a model on a specific task

- but can lead to reduction in ability on other task.

- How to avoid catastrophic forgetting

- First note that you may not have to.
- Fine tune multiple tasks at the same time.
- Consider Parameter Efficient Fine tuning (PEFT)

* Multi-task instruction fine tuning

- multi-task fine tuning is an extension of single task fine tuning where training data is comprised of example input & outputs for multiple tasks.

- one ~~disad~~ drawback of multi task fine tuning is you will require a lot of data. you may need as many as 50-100,000 examples in your training set.

- For ex. FLAN family of models are trained using multi-task fine tuning.

classmate
Date
Page

* Model Evaluation.

- LLM Evaluation - challanges

- Since LLM works on Natural Languages it is hard to evaluate LLMs.

- LLM Evaluation - metrics.

ROGUE

- Used for text summarization
- Compare a summary to one or more reference summaries

BLEU SCORE

- Used for text translation
- Compares to human generated translations.

unigram → one word
bi gram → two words
n-gram → group of n-words

ROGUE 1 Recall = $\frac{\text{unigram matches}}{\text{unigram in reference.}}$

ROGUE 1 Precision = $\frac{\text{unigram matches}}{\text{unigrams in output}}$

ROGUE 1 F1 = $2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

- Similar to ROGUE, we can calculate ROGUE 2 with help of bi-grams

- ROGUE-L, where L is Longest common subsequence (LCS)

{ Longet common subsequence (LCS) is the longest sequence of words that appear in both sentences is same order, but not neccessarily, consicitewe }

- ROGUE clipping.

BLEU SCORE:

- The BLEU SCORE quantifies the quality of translation by checking how many n-grams in machine-generated translation match those in reference translation

- BLEU(metric) = Avg(Precision across range of ~~metric~~) n-gram sizes

* Benchmarks

- In order to evaluate LLMs more holistically, we can make use of pre-existing datasets & associated benchmarks.

- Benchmarks such as ~~GLEU~~ GLUE, SuperGLUE & HELM, cover a wide range of task scenarios.

- Benchmarks for massive models.

- massive multitask Language Understanding (MMLU) 2021

- Big-bench, Big-bench hard, Big-bench lite 2022

- ~~Holisti~~

Holistic Evaluation of Language models (HELM)

Metrics
- Accuracy
- Callibration
- Robustness
- Fairness
- Bias
- Toxicity
- Efficiency.

classmate
Date
Page

* Parameter Efficient fine-tuning

- PEFT only updates small set of parameters
- In some cases only 15-20% of original LLM weights are trained & rest are frozen.
- Less prone to Castostropic forgetting.
- PEFT fine tuning saves space & is flexible.

• PEFT Tradeoffs

- Parameter Efficiency
- Memory Efficiency
- model performance.
- Inference Costs
- Training speed.

- PEFT methods

i) Selective methods

- Selective methods are those that fine tune only a subset of of the original LLM parameters.

- There are several approaches that you can take to identify which parameters you want to update.

- you have option to train only certain components of the mode or specific layers or even individual parameter types.

- Researchers have found that performance of this methods is mixed & there are significant tradeoffs between parameter efficiency & compute efficiency.

ii) Reparameterization methods

- Reparametirization methods also work with original LLM parametors, but reduce the number of parameters to train by creating new low rank transformations of the original network weights.

- A commonly used technique of this type is LORA

iii) Additive Methods

- Additive methods carry out fine tuning by freezing all the original LLM weights & introducing new trainable components.

- Here there are two main approaches

  - Adapters
  - self prompts.

- Adapters → Add new trainable layers to the architecture of the model, typically inside the encoder or decoder components. after the attention or feed-forward layers.

- Self prompters → Keeps the model architecture fixed or frozen. & focus on manipulating input to achieve better result. This can be done by adding trainable parameters to prompt embedding or keeping the input fixed & retraining embeddings weights.

classmate
Date
Page

* PEFT Techniques 1: LoRA

LoRA → Low Rank Adaptation of Large Language Models

- Freeze most of the original LLM weights
- Inject 2 rank decomposition matrix.
- Train the weights of the smaller matrices

Steps to update model for inference.

- Matrix multiply the 2 low rank matrices

$B * A = BXA$

- Add to original weights.

Frozen weights + BXA

For ex:

Use the base transformer model presented in "Attention is all you need" paper

- Transformer weights have dimensions $d \times k = 512 \times 64$
- So $512 \times 64 = 32,768$ trainable parameters

In LoRA rank $r = 8$

- A has dimensions $r \times k = 8 \times 64 = 512$
- B has dimensions $d \times r = 512 \times 8 = 4,096$ trainable parameters
- 86% reduction in parameters to train.

You can use LoRA to train for many tasks. Switch out the weights when you need them & avoid storing multiple full sized versions of LLM.

classmate
Date
Page

How to choose the rank of LoRA

↳ This is still a area of research. In theory lower the rank, the smaller the number of trainable parameters.

* PEFT Techniques 2: soft prompts

- prompt tuning adds trainable "soft prompt" to inputs.
- weights of model & soft prompt trained
- Insted embedding veeters of model gets updated.
- Very parameter Efficient statergy.
- performance of prompt tuning.
- prompt tuning doesn't perform ~~well~~ as well as ~~fin~~ full fine-tuning for smaller LLMs.
- But as model size reaches 10B param prompt tuning can be as efficient as full fine-tuning.