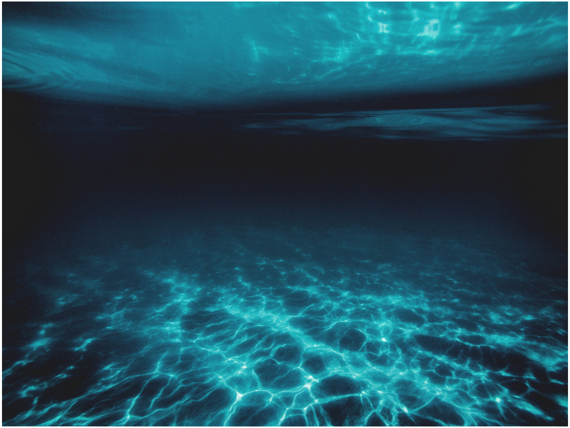
Natural Language Processing

Lecture 6 : Post-training

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture partially based on material originally developed by :
Elena Voita, University of Edinburgh

Post-training



Jonathan Borba from Usplash

Post-training

To make practical use of pretrained LLMs, we need to

- interface these models with downstream applications
- adapt these models to a specific domain of interest

This process is called **post-training** or **adaptation**. It is the prevalent paradigm today in natural language processing.

Post-training typically uses supervised learning (labeled data) for the task of interest.

Post-training

Post-training groups together several techniques

- supervised fine-tuning
- instruction-tuning
- alignment

Post-training typically uses much less computational resources than pretraining, and does not target all of the parameters of the LLM.

Several **efficient** techniques are known for parameter updating during post-training.

Fine-tuning

Joel Wyncott from Unsplash

Fine-tuning

We might want to specialize a LLM for a new task or a new domain that has not appeared sufficiently in the pre-training data.

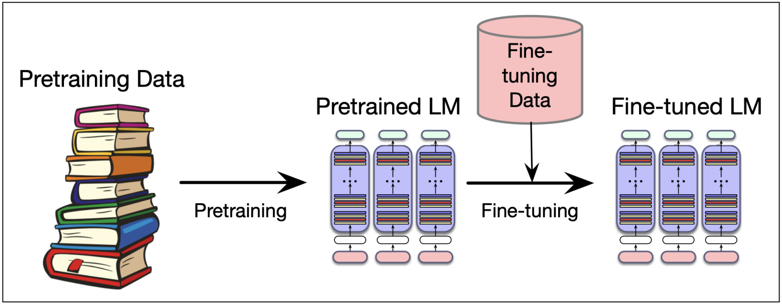
The standard approach is **fine-tuning**

- add a head layer for the task at hand, at the top of the transformer, with learnable parameters
- make (possibly minimal) adjustments to the pretrained parameters of the LLM

In contrast with fine-tuning, retraining the whole LLM (all parameters) results in so-called **catastrophic forgetting**.

Fine-tuning

A pre-trained model can be fine-tuned to a particular domain, dataset, or task.



Fine-tuning

Example : Under BERT pre-trained model, let $\mathbf{z}_{\text{CLS}}$ be the embedding for the token `[CLS]`. For **sentiment analysis** define the classifier

$$\mathbf{y}_{\text{CLS}} = \text{softmax}(\mathbf{W}_C \cdot \mathbf{z}_{\text{CLS}} + \mathbf{b})$$

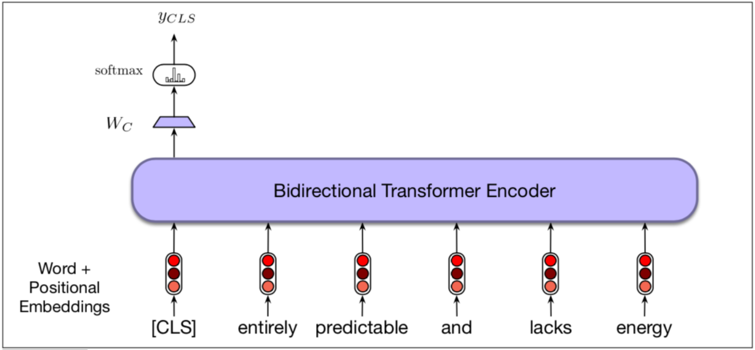
Use labeled data to learn $\mathbf{W}_C, \mathbf{b}$ and to **update** the weights for the pre-trained language model itself.

In practice, only **minimal** changes to the language model parameters are needed, limited to updates over the final few layers of the transformer.

We will see later efficient techniques for updating the pretrained parameters of the LLM.

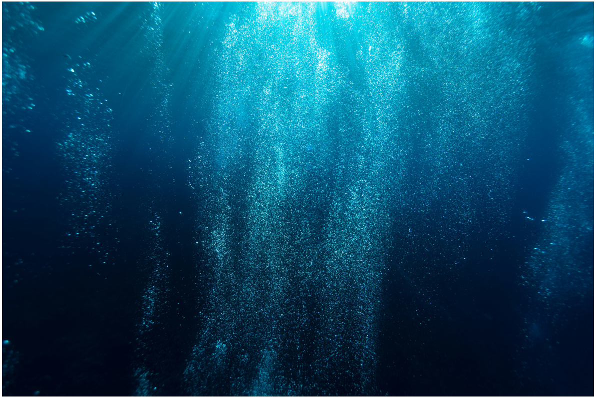
Fine-tuning

Example : In sentiment analysis we learn $\mathbf{W}_C, \mathbf{b}$ and update only top-most layers of the transformer.



W_C is usually called the **classification head**.

Instruction tuning



Sarah Lee from Unsplash

LLM have not been designed to answer questions or to follow instructions, even in presence of strong signals such as prompt. Their only objective is to predict the next word.

Example :

Prompt: Explain the moon landing to a six year old in a few sentences.
Output: Explain the theory of gravity to a 6 year old.

Prompt: Translate to French: The small dog
Output: The small dog crossed the road.

Instruction tuning

Instruction tuning is a method for making an LLM better at following instructions.

The training data consists of

- instructions and answers for a wide range of tasks, and/or
- questons and answers in a specific domain of interest

We continue training the model with the guess-the-next-token objective, **limited to** the answer part of each example

In instruction tuning, the training corpus is simply treated as additional training data.

This is often referred to as continual learning.

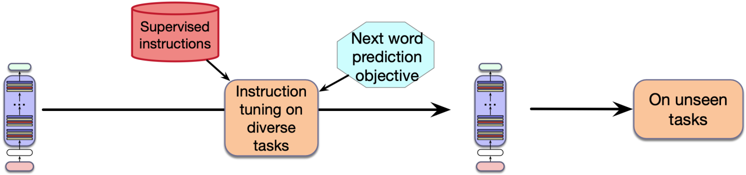
Instruction tuning

Instruction tuning is a form of **supervised** learning, because each instruction or question in the training data has a supervised objective: a correct answer, or a response to the instruction.

Instruction tuning is also called **supervised fine tuning** (SFT).

Instruction tuning

Instruction tuning (supervised fine tuning) viewed as a second pass of language modeling training.



Instruction tuning

Instruction tuning datasets can be created

- manually, through crowdworkers based on carefully written annotation guidelines
- by adapting existing datasets
- automatically, using powerful chatBots and prompts based on human written guidelines

Instruction tuning

Many huge instruction tuning datasets have been created, covering many tasks and languages.

Most popular

- **SuperNatural Instructions**: 12 million examples from 1600 tasks
- **Flan**: see next slides
- **Aya**: 503 million instructions in 114 languages from 12 tasks

FLAN

Fine-tuned Language Net (FLAN) compiles several datasets into a mix of zero-shot, few-shot and chain-of-thought templates. It is a specific set of instructions used to fine-tune different models.

Consists of 473 datasets across 146 task categories.

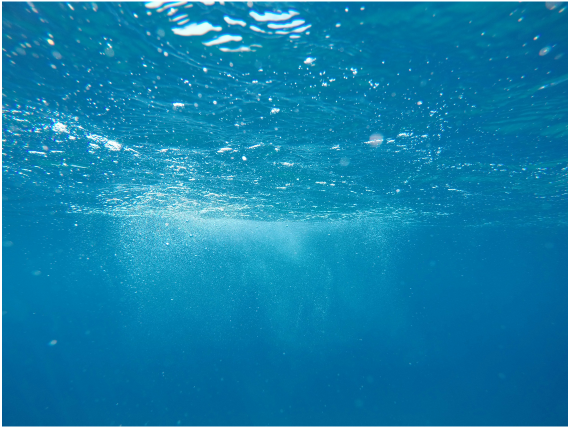
Several LLM instruction tuned with FLAN: Flan-T5, Flan-PaLM, etc.

FLAN

		Model Details				Data Collection & Training Details			
Release	**Collection**	**Model**	**Base**	**Size**	**Public?**	**Prompt Types**	**Tasks in Flan**	**# Exs**	**Methods**
2020 05	UnifiedQA	UnifiedQA	RoBerta	110-340M	P	ZS	46 / 46	750k	
2021 04	CrossFit	BART-CrossFit	BART	140M	NP	FS	115 / 159	71.M	
2021 04	Natural Inst v1.0	Gen. BART	BART	140M	NP	ZS / FS	61 / 61	620k	+ Detailed k-shot Prompts
2021 09	Flan 2021	Flan-LaMDA	LaMDA	137B	NP	ZS / FS	62 / 62	4.4M	+ Template Variety
2021 10	P3	T0, T0+, T0++	T5-LM	3-11B	P	ZS	62 / 62	12M	+ Template Variety + Input Inversion
2021 10	MetaICL	MetaICL	GPT-2	770M	P	FS	100 / 142	3.5M	+ Input Inversion + Noisy Channel Opt
2021 11	ExMix	ExT5	T5	220M-11B	NP	ZS	72 / 107	500k	+ With Pretraining
2022 04	Super-Natural Inst.	*Tk*-Instruct	T5-LM, mT5	11-13B	P	ZS / FS	1556 / 1613	5M	+ Detailed k-shot Prompts + Multilingual
2022 10	GLM	GLM-130B	GLM	130B	P	FS	65 / 77	12M	+ With Pretraining + Bilingual (en, zh-cn)
2022 11	xP3	BLOOMz, mT0	BLOOM, mT5	13-176B	P	ZS	53 / 71	81M	+ Massively Multilingual
2022 12	Unnatural Inst.†	T5-LM-Unnat. Inst.	T5-LM	11B	NP	ZS	~20 / 117	64k	+ Synthetic Data
2022 12	Self-Instruct†	GPT-3 Self Inst.	GPT-3	175B	NP	ZS	Unknown	82k	+ Synthetic Data + Knowledge Distillation
2022 12	OPT-IML Bench†	OPT-IML	OPT	30-175B	P	ZS + FS CoT	~2067 / 2207	18M	+ Template Variety + Input Inversion + Multilingual
2022 10	Flan 2022 (ours)	Flan-T5, Flan-PaLM	T5-LM, PaLM	10M-540B	P NP	ZS + FS CoT	1836	15M	+ Template Variety + Input Inversion + Multilingual

The Flan Collection: Designing Data and Methods for Effective Instruction Tuning, Longpre et al. 2023

Model Alignment



Krystian Tambur from Unsplash

Model Alignment

SFT results in LLMs that generate coherent and contextually appropriate responses.

However, some responses may contain harmful, unethical, socially biased, and negative, or even illegal content.

Alignment is a post-training process ensuring that LLMs meet human expectations, ethical standards, and personalized needs.

Model Alignment

Reinforcement Learning (RL) has emerged as the most popular approach for model alignment, empowering LLMs toward delivering human-like responses.

RL methods

- exploit a **reward function** trained on some dataset of response rankings
- apply the reward function to train an LLM

RL methods lay in between supervised and unsupervised learning.

Model Alignment

Reinforcement Learning from Human Feedback (RLHF) involves

- fitting of a reward model to a dataset of human preferences
- training an LLM to generate responses with high reward

Model Alignment

Reward function is trained on high-quality **preference datasets**, consisting in two contrastive answers to some question

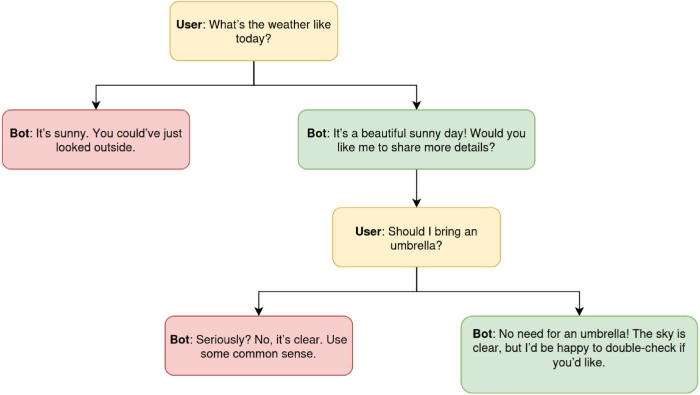
- a **preferred** answer, meeting the desired standard
- a **dispreferred** answer, whose style should be discouraged

Preference datasets are often written by humans, or else a combination of human and very powerful genAI models.

Alignment is also called **preference finetuning**.

Model Alignment

Example : preference data for conversation task.



Model Alignment

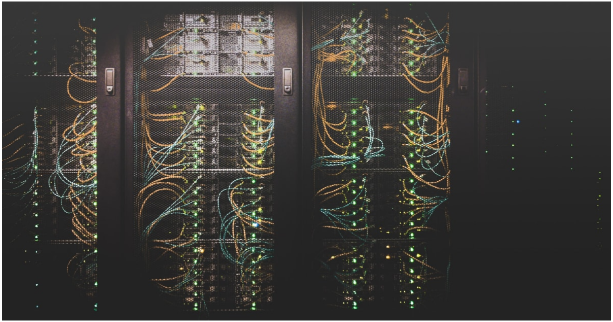
Two most popular RLHF algorithms.

Proximal policy optimization (PPO), first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM to maximize the reward without drifting too far from the original model.

Problems: sensitivity to hyperparameters, inherent instability during training.

Direct preference optimization (DPO), enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss.

Parameter efficient fine-tuning



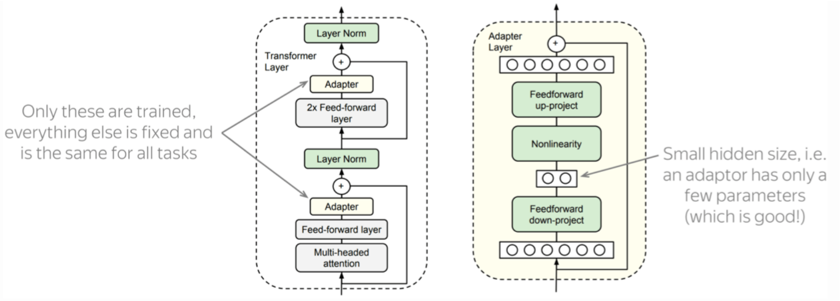
Adapters

When working with huge pre-trained models, instruction tuning and fine tuning may still be **inefficient**.

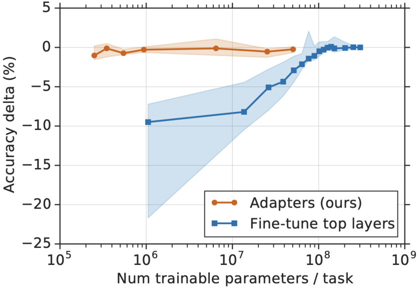
Alternatively, one could fix the pre-trained model, and train only small, very simple components called **adapters**.

With adapter modules post-training becomes very **efficient**: the largest part of the pre-trained model is shared between all downstream tasks.

Transformer with adapter module consisting of a two-layer feed-forward network with a nonlinearity and a residual connection.



Adapter-based tuning attains a similar performance to top-layer fine-tuning, with two orders of magnitude fewer trained parameters.



Houlsby *et al.*, 2019

LoRA

LoRA (Low-Rank Adaptation) is a popular fine-tuning strategy, alternative to adapters. It drastically reduces the number of trainable parameters.

LoRA is based on the idea of **intrinsic dimensions**: there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space.

LoRA is known for its high efficiency, and for avoiding catastrophic forgetting.

LoRA

We can think of **weight update** as follows

$$W \quad \leftarrow \quad W + \Delta W$$

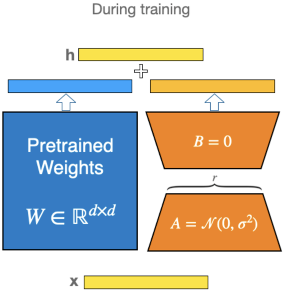
where $W \in \mathbb{R}^{d \times d}$.

In LoRA we update the weights using a decomposition of ΔW into two low-rank matrices

$$\Delta W \quad = \quad BA$$

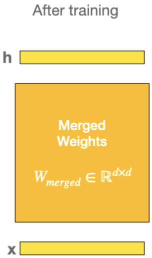
where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times d}$, and r is some low **rank**.

LoRA



$$h = Wx + BAx$$

$$h = \underbrace{(W + BA)}_{W_{merged}}x$$



Saurav Maheshkar, A Brief Introduction to LoRA: Low-Rank Adaptation of Large Language Models

Discussion

LLMs learn very powerful general representation about language and real world.

Traditional approach in NLP is to develop application from scratch.

Fine-tune approach: use LLM as a basis, and fine-tune it to the desired task — think about LLM as multi-task models!

In-context approach: provide a few examples of the task, and ask the question/instance at hand — prompt engineering.

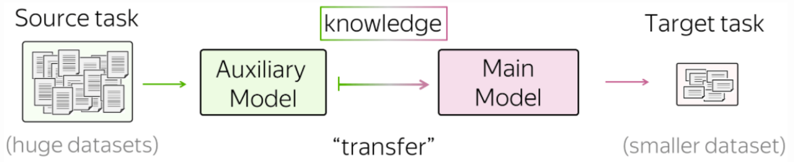
We introduce **prompt engineering** in next lecture.

Rhys Moult from Unsplash

The pre-training/fine-tuning paradigm is a special case of a machine learning approach called transfer learning.

The general idea of **transfer learning** is to reuse information from a previously learned source task for the learning of a target tasks.

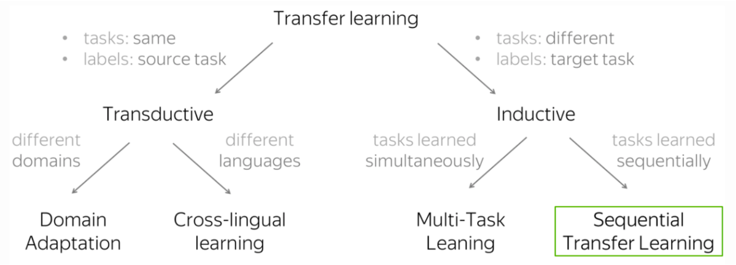
This is used when you don't have enough data for the target task.



There are several types of transfer learning

See for instance Sebastian Ruder's blog post at

https://ruder.io/state-of-transfer-learning-in-nlp/



Fine-tuning, also known as **sequential transfer learning**, is the most popular transfer learning approach in NLP.

Research papers

Emil Widlund on Unsplash

Title: Direct Preference Optimization: Your Language Model is Secretly a Reward Model understanding?
Authors: Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn
Source: 29 Jul 2024
Content: This paper introduces a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm is stable, performant, and computationally lightweight.

https://arxiv.org/abs/2305.18290