

towards
data science

The world's leading publication for data science, AI, and ML professionals.

Sign in

Contributor Portal

LATEST EDITOR'S PICKS DEEP DIVES CONTRIBUTE | NEWSLETTER



Octopus Deploy

Deploy more often and with confidence

Automate and streamline releases with Octopus Deploy. Boost team productivity.

LARGE LANGUAGE MODELS

Different ways of training LLMs

And why prompting is none of them

Dorian Drost

Jul 21, 2023 • 13 min read



In the realm of large language models (LLMs), there is a variety of training mechanisms with different means, requirements, and goals. As they serve different purposes, it is important to not confuse them with each other and to be aware of the different scenarios they are applicable to.

In this article, I want to give an overview of some of the most important training mechanisms which are *pretraining*, *finetuning*, *reinforcement learning from human feedback (RLHF)*, and *adapters*. Additionally, I will discuss the role of *prompting*, which is not considered to be a learning mechanism per se, and shed some light on the concept of *prompt tuning*, which builds a bridge between prompting and actual training.

Pretraining



"Train". Like in "Training". I'm sorry...Photo by [Brian Suman](#) on [Unsplash](#)

Pretraining is the most fundamental way of training and is equal to what you might know as training from other machine learning domains. Here you start with an untrained model (i.e. one that has randomly initialized weights) and train to predict the next token given a sequence of previous tokens. To this end, a large corpus of sentences is collected from various sources and given to the model in small chunks.

The training mode used here is called *self-supervised*. From the perspective of the model being trained, we can speak of a supervised learning approach, because the model always gets the correct answer after having made a prediction. For example, given the sequence *I like ice ...* the model might predict *cones* as the next word and may then be told that the answer is wrong because the actual next word was *cream*. Eventually, the loss can be calculated and the model weights are adapted to predict better next time. The reason to call it *self-supervised* (instead of simply *supervised*) is, that there is no need to collect the labels in an expensive procedure beforehand, but they are already included in the data. Given the

sentence *I like ice cream*, we can automatically split it into *I like ice* as the input and *cream* as the label, which requires no human effort. Although it is not the model itself, that does that, it is still performed automatically by the machine, hence the idea of an AI supervising *itself* in the learning process.

Eventually, training on large amounts of text, the model learns to encode the structure of language in general (e.g. it learns, that *I like* could be followed by a noun or a participle) as well as the knowledge that is included in the texts it saw. For example, it learned, that the sentence *Joe Biden is ...* is often followed by *the president of the United States*, and hence has a representation of that piece of knowledge.

This pretraining has already been done by others and you can use models such as GPT out of the box. Why should you ever train a similar model, though? Training a model from scratch can become necessary if you work with data that has properties similar to language, but that is not common language itself. Musical notation could be an example, that is somehow structured like a language. There are certain rules and patterns about which pieces may follow each other, but an LLM that is trained on natural language can't handle that kind of data, so you would have to train a new model. However, the architecture of LLMs may be suitable, due to the many similarities between musical notations and natural language.

Finetuning





Those knobs are used to finetune a string instrument. Photo by [Tony Woodhead](#) on [Unsplash](#)

Although a pretrained LLM is, due to the knowledge it encodes, able to perform a various number of tasks, there are two main shortcomings in it, which are the structure of its output and the absence of knowledge that wasn't encoded in the data in the first place.

As you know, an LLM always predicts the next tokens given a sequence of tokens before. For continuing a given story that might be fine, but there are other scenarios where this is not what you want. If you need a different output structure, there are two main ways to make that possible. You can either write your prompts in such a way, that the model's inert ability to predict the next tokens solves your task (which is called *prompt engineering*), or you change the last layer's output such that it reflects your task as you would do in any other machine learning model. Think of a classification task, where you have N of classes. With prompt engineering, you could instruct your model to always output the classification label after a given input. With finetuning, you could change the last layers to have N output neurons and derive the predicted class from the neuron with the highest activation.

The other limit of the LLM lies in the data it was trained with. As the data sources are quite rich, the most known LLMs encode a large variety of common knowledge. Thus they can tell you, among others, about the president of the United States, Beethoven's major works, the fundamentals of quantum physics, and Sigmund Freud's major theories. However, there are domains that the models don't know about, and if you need to work with such domains, finetuning might be relevant for you.

The idea of finetuning is to take an already pretrained model and continue its training with different data, and by changing only the weights in the last layers during the training. This requires only a fraction of the resources that were needed in the initial training and

hence can be performed much faster. On the other hand, the structures the model learned during the pretraining are still encoded in the first layers and can be utilized. Say you want to teach your model about your favorite, but less-known fantasy novels, that have not been part of the training data. With finetuning, you leverage the model's knowledge about natural language in general to make it understand the new domain of fantasy novels.

RLHF Finetuning



RLHF finetuning is about maximizing the reward. Photo by [Alexander Grey](#) on [Unsplash](#)

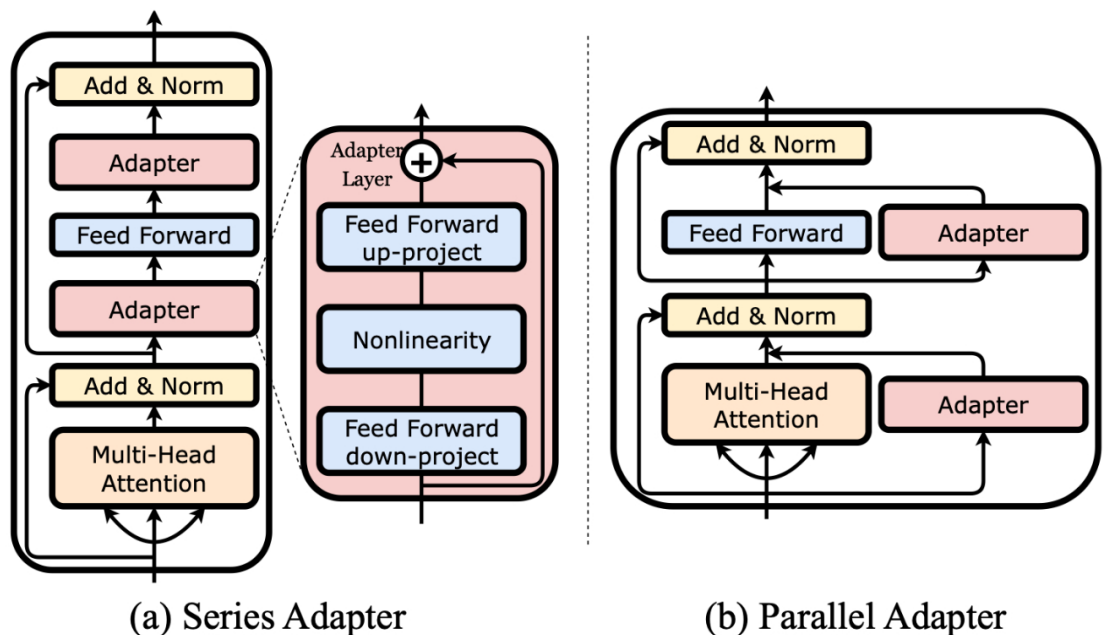
A special case of finetuning the model is reinforcement learning from human feedback (RLHF), which is one of the main differences between a GPT model and a chatbot like Chat-GPT. With this kind of finetuning, the model is trained towards producing outputs a human finds most useful in their conversation with the model.

The main idea is as follows: Given an arbitrary prompt, multiple outputs from the model are generated for that prompt. A human ranks those outputs according to how useful or appropriate they find them. Given four samples A, B, C, and D, the human might decide that C is the best output, B is a little worse but equal to D, and A is the worst output for that prompt. That would result in an ordering of C > B = D > A.

the worst output for that prompt, that would result in an order $C > B = D > A$. Next, this data is used to train a reward model. That is an entirely new model that learns to rate an LLM's outputs by giving it a reward that reflects the human's preferences. Once the reward model is trained, it can substitute the human in that produce. Now the outputs from the model are rated by the reward model, and that reward is given as feedback to the LLM that is then adapted to maximize the reward; an idea very similar to that of GANs.

As you see, for this kind of training human-labeled data is needed, which requires quite some effort. However, the amount of data needed is finite, as the idea of the reward model is to generalize from that data so that it can rate the llm on its own, once it has learned its part. RLHF is often used to make LLM outputs more conversation-like or avoid undesired behavior such as a model being mean, intrusive, or insulting.

Adapters



Two kinds of adapters to be plugged into an already existing network. Image taken from <https://arxiv.org/pdf/2304.01933.pdf>.

In the aforementioned finetuning, we adapt some of the model's parameters in the last layers, while the other parameters in the previous layers stay as they are. There is an alternative to that, though, that promises more efficiency by a smaller number of parameters required to train, which is called adapters.

parameters required to train, which is called adapters.

Using adapters means adding additional layers to an already trained model. During the finetuning, only those adapters are trained while the rest of the model's parameters is not changed at all. However, those layers are much smaller than the layers the model comes with, which makes it easier to tune them. Additionally, they can be inserted at different positions in the model, not only at the very end. In the picture above you see two examples; one, where an adapter is added as a layer in a serial fashion, and one where it is added in parallel to an already existing layer.

Prompting



Prompting is more about telling a model what to do, not how to do it. Photo by [Ian Taylor](#) on [Unsplash](#)

You may wonder if prompting counts as another way of training a model. Prompting means constructing the instructions that precede the actual model's input, and especially if you use few-shot-prompting, you provide examples to the LLM within the prompt, which is very similar to training, which also consists of examples that are presented to the model. However, there are reasons why prompting is something else than training a model. First of all, from the plain definition, we only speak of training if weights are updated.

and that is not done during prompting. When creating a prompt, you don't change any model, you change no weights, you produce no new model and you also don't change the knowledge or representation encoded in the model. Prompting should rather be seen as a way of instructing an LLM and telling it what you want from it. Consider the following prompt as an example:

```
""""Classify a given text regarding its sentiment.
```

```
Text: I like ice cream.
```

```
Sentiment: negative
```

```
Text: I really hate the new AirPods.
```

```
Sentiment: positive
```

```
Text: Donald is the biggest jerk on earth. I hate him so much!
```

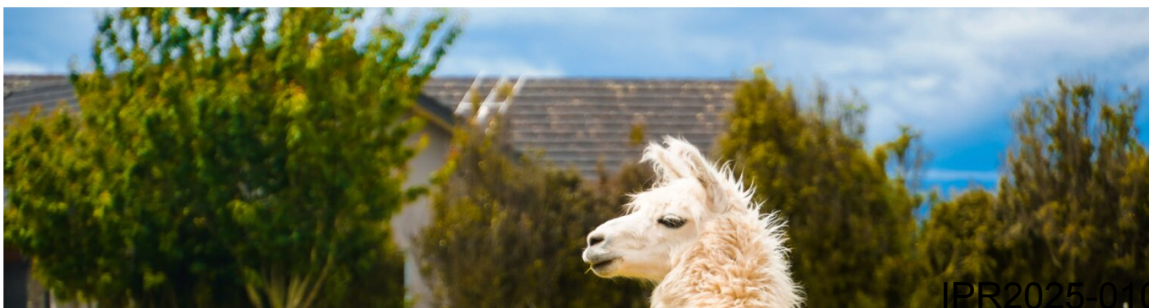
```
Sentiment: neutral
```

```
Text: {user_input}
```

```
Sentiment: """"
```

I instructed the model to do a sentiment classification, and, as you might have noticed, the examples I gave to the model are all wrong! If a model was trained with such data, it would mix up the labels *positive*, *negative*, and *neutral*. What happens now, if I ask the model to classify the sentence *I like ice cream*, which was part of my examples? Interestingly, it classifies it as *positive*, which is contrary to the prompt, but correct on a semantic level. That is because the prompt didn't train the model and didn't change the representation of what it has learned. The prompt simply informs the model about the structure I expect, namely that I expect the sentiment label (which can be *positive*, *negative*, or *neutral*) to follow after the colon.

Prompt tuning





Prompt tuning is also called soft prompting. Soft like alpaca whool...Photo by [Advocator SY](#) on [Unsplash](#)

Although a prompt itself is not training the llm, there is a mechanism called *prompt tuning* (also called *soft prompting*), that is related to prompting and can be seen as a kind of training.

In the previous example, we considered the prompt to be a natural language text that is given to the model in order to tell it what to do and that precedes the actual input. That is, the model input becomes , so, for example, *<label the following a positive, negative, or neutral:>* . When creating the prompt by ourselves, we speak of *hard prompting*. In *soft prompting*, the format will be kept, but the prompt itself is not designed by ourselves but learned with data. In detail, the prompt is consisting of parameters in a vector space, and those parameters can be tuned during training to obtain a smaller loss and hence better answers. That is, after training the prompt will be that sequence of characters that leads to the best answers for our given data. The model parameters, however, are not trained at all.

A big advantage of prompt tuning is, that you can train multiple prompts for different tasks but still use them with the same model. Just like in hard prompting, where you might construct one prompt for text summarization, one for sentiment analysis, and one for text classification, but use all with the same model, you can tune three prompts for that purposes and still use the same model. If you would have used finetuning, on the contrary, you would have ended up with three models that serve only their specific task each.

Summary

We just saw a variety of different training mechanisms, so let's do a short summary at the end.

- *Pretraining an LLM means teaching it to predict the next token in a self-supervised fashion.*
- *Finetuning is adapting the weights of a pretrained LLM in the last layers and can be used to adapt the model to a specific context.*
- *RLHF aims at adapting a model's behavior to match human expectations and needs additional labeling effort.*
- *Adapters allow for a more efficient way of finetuning due to small layers, that are added to the pretrained LLM.*
- *Prompting is not considered training per se, because it doesn't change the model's internal representation.*
- *Prompt tuning is a technique to tune weights that produce a prompt but doesn't affect the model weights themselves.*

Of course, there are many more training mechanisms out there, with new ones being invented every day. LLMs can do much more than just predicting text, and teaching them to do so requires a variety of skills and techniques, some of which I just presented to you.

Further reading

Instruct-GPT is one of the most famous examples of RLHF:

- <https://openai.com/research/instruction-following>.
- *Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.*

An overview of common forms of adapters can be found in the LLM-Adapters project:

- <https://github.com/AGI-Edgerunners/LLM-Adapters>
- *Hu, Z., Lan, Y., Wang, L., Xu, W., Lim, E. P., Lee, R. K. W., ... & Poria, S. (2023). LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. arXiv preprint arXiv:2304.01933.*

Prompt Tuning has been explored here:

- *Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691.*

Some nice explanations of prompt tuning can be found here:

Some nice explanations of prompt tuning can be found here.

- https://huggingface.co/docs/peft/conceptual_guides/prompting.
- <https://ai.googleblog.com/2022/02/guiding-frozen-language-models-with.html>

Like this article? [Follow me](#) to be notified of my future posts.

• • •

WRITTEN BY

Dorian Drost

See all from Dorian Drost >

Topics:

- Ai Training
- Editors Pick
- Large Language Models
- Llm
- Prompt Tuning

Share this article:   

Related Articles

ARTIFICIAL INTELLIGENCE

What Do Large Language Models “Understand”?

A deep dive on the meaning of understanding and how it applies to LLMs

Tarik Dzekman

August 21, 2024 • 31 min read



ARTIFICIAL INTELLIGENCE

Making Sense of the Promise

IPR2025-01086

(and Risks) of Large Language Models

Our weekly selection of must-read Editors' Picks and original features

TDS Editors

April 27, 2023 • 4 min read



DATA SCIENCE

The Creative, Occasionally Messy World of Textual Data

Our weekly selection of must-read Editors' Picks and original features

TDS Editors

November 16, 2023 • 3 min read



LARGE LANGUAGE MODELS

GPT and Beyond: The Technical Foundations of LLMs

Our weekly selection of must-read Editors' Picks and original features

TDS Editors

August 3, 2023 • 3 min read

LARGE LANGUAGE MODELS

Deep Dive into LLaMA 3 by Hand

Explore the nuances of the transformer architecture behind Llama 3 and its prospects for the...

Srijanie Dey, PhD

May 3, 2024 • 12 min read



LARGE LANGUAGE MODELS

Deep Dive into Transformers by Hand

Explore the details behind the power of transformers

Srijanie Dey, PhD

April 12, 2024 • 6 min read



DATA SCIENCE

IPR2025-01086

Orca Security EX1024 Page 12

Deep Dive into Sora's Diffusion Transformer (DiT) by Hand 📖

Explore the secret behind Sora's state-of-the-art videos

Srijanie Dey, PhD

April 2, 2024 • 13 min read



towards
data science

Your home for data science and AI. The world's leading publication for data science, data analytics, data engineering, machine learning, and artificial intelligence professionals.

© Insight Media Group, LLC 2025

[Subscribe to Our Newsletter >](#)

[ABOUT](#) · [ADVERTISE](#) · [PRIVACY POLICY](#) ·

[TERMS OF USE](#)

[COOKIES SETTINGS](#)