The self-attention weights that are learned during training and stored in these layers reflect the importance of each word in that input sequence to all other words in the sequence. But this does not happen just once, the transformer architecture actually has multi-headed self-attention. This means that multiple sets of self-attention weights or heads are learned in parallel independently of each other. The number of attention heads included in the attention layer varies from model to model, but numbers in the range of 12-100 are common. The intuition here is that each self-attention head will learn a different aspect of language. For example, one head may see the relationship between the people entities in our sentence. Whilst another head may focus on the activity of the sentence. Whilst yet another head may focus on some other properties such as if the words rhyme.

In original Transformer architecture there is 512 dimension of vector embedding’s are used

For Encoder only models, the sequence to sequence tokens would of same length only without modification. Like Bert if used for sentimental analysis which is achieved by some modification in architecture

"Attention is All You Need" is a research paper published in 2017 by Google researchers

The Transformer architecture consists of an encoder and a decoder, each of which is composed of several layers. Each layer consists of two sub-layers: a multi-head self-attention mechanism and a feed-forward neural network. The multi-head self-attention mechanism allows the model to attend to different parts of the input sequence, while the feed-forward network applies a point-wise fully connected layer to each position separately and identically.

The Transformer model also uses residual connections and layer normalization to facilitate training and prevent over fitting. In addition, the authors introduce a positional encoding scheme that encodes the position of each token in the input sequence, enabling the model to capture the order of the sequence without the need for recurrent or convolutional operations.

You may have to revise the language in your prompt or the way that it's written several times to get the model to behave in the way that you want. This work to develop and improve the prompt is known as prompt engineering.

Greedy decoding. This is the simplest form of next-word prediction, where the model will always choosethe word with the highest probability.This method can work very wellfor short generation but issusceptible to repeated wordsor repeated sequences of words.If you want to generate text that's more natural,more creative and avoids repeating words,you need to use some other controls.Random sampling is the easiest wayto introduce some variability.Instead of selecting the most probable wordevery time with random sampling,the model chooses an output word at random usingthe probability distribution to weight the selection

 temperature. This parameter influences the shape of the probability distribution that the model calculates for the next token. Broadly speaking, the higher the temperature, the higher the randomness, and the lower the temperature, the lower the randomness.

The Vocareum AI Notebook is a cloud-based, Jupyter-powered platform that integrates with your LMS, providing a ready-to-use environment with essential data science libraries, GenAI technologies, and GPU resources for enhanced AI education.

Researchers have found that applying

LoRA to just the self-attention layers of

the model is often enough to

fine-tune for a task and achieve performance gains.

However, in principle,

you can also use LoRA on

other components like the feed-forward layers.

you can often perform this method of

parameter efficient fine tuning with

a single GPU and avoid

the need for a distributed cluster of GPUs.

Since the rank-decomposition matrices are small,

you can fine-tune a different set for each task and then

switch them out at inference time

by updating the weights.

Suppose you train a pair of

LoRA matrices for a specific task;

let's call it Task A.

To carry out inference on this task,

you would multiply these matrices together and

then add the resulting matrix

to the original frozen weights.

You then take this new summed weights matrix

and replace the original weights

where they appear in your model.

You can then use this model to

carry out inference on Task A.

If instead, you want to carry out

a different task, say Task B,

you simply take the LoRA matrices you

trained for this task, calculate their product,

and then add this matrix to

the original weights and update the model again

You might be wondering how to choose

the rank of the LoRA matrices.

This is a good question and

still an active area of research

The soft prompt vectors have the same length as the embedding vectors of

the language tokens.

And including somewhere between 20 and

100 virtual tokens can be sufficient for good performance.

The tokens that represent natural language are hard in the sense that they

each correspond to a fixed location in the embedding vector space.

However, the soft prompts are not fixed discrete words of natural language.

Instead, you can think of them as virtual tokens that can take on any

value within the continuous multidimensional embedding space.

And through supervised learning, the model learns the values for

these virtual tokens that maximize performance for a given task.

prompt tuning, the weights of the large language model

are frozen and the underlying model does not get updated.

Instead, the embedding vectors of the soft prompt gets updated over time to

optimize the model's completion of the prompt.

similar to what you saw with LoRA.

You can train a different set of soft prompts for each task and

then easily swap them out at inference time.

You can train a set of soft prompts for one task and a different set for another.

To use them for inference, you prepend your input prompt with

the learned tokens to switch to another task, you simply change the soft prompt

combine LoRA with the quantization techniques you learned

about in week 1 to further reduce your memory footprint.

This is known as QLoRA in practice

Let's call this intermediate version of

the model the RL updated LLM.

These series of steps together

forms a single iteration of the RLHF process.

These iterations continue for a given number of epics,

similar to other types of fine tuning. You can also define a maximum number of steps,

for example, 20,000 as the stopping criteria

PPO also ensures to keep the model updates within

a certain small region called the trust region

The PPO policy objective

is the main ingredient of this method.

Remember, the objective is to find

a policy whose expected reward is high.

In other words, you're trying to

make updates to the LLM weights that

result in completions more aligned with

human preferences and so receive a higher reward.

The policy loss is the main objective that

the PPO algorithm tries to optimize during training.

The advantage term tells

you how better or worse the current

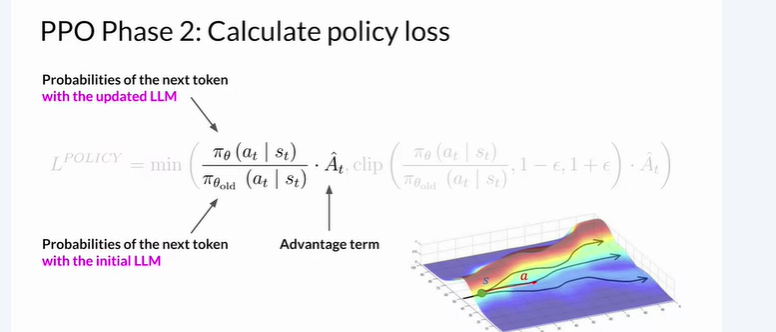
token A\_t is with respect to all the possible tokens.

In this visualization,

the top path which goes higher is better completion,

receiving a higher reward.

The bottom path goes down which is a worst completion



While the policy loss moves

the model towards alignment goal,

entropy allows the model to maintain creativity.

Q-learning is

an alternate technique for fine-tuning LLMs through RL,

but PPO is currently the most popular method.

In my opinion, PPO is popular because it has

the right balance of complexity and performance

researchers at Stanford published a paper

describing a technique

called direct preference optimization,

which is a simpler alternate to RLHF

KL-Divergence, or Kullback-Leibler Divergence, is a concept often encountered in the field of reinforcement learning, particularly when using the Proximal Policy Optimization (PPO) algorithm. It is a mathematical measure of the difference between two probability distributions, which helps us understand how one distribution differs from another. In the context of PPO, KL-Divergence plays a crucial role in guiding the optimization process to ensure that the updated policy does not deviate too much from the original policy.

A library that you can use to train transformer language models with reinforcement learning, using techniques such as PPO, is TRL (Transformer Reinforcement Learning

Constitutional AI is one approach of scale supervision.

First proposed in 2022 by researchers at Anthropic,

Constitutional AI is a method for training models using

a set of rules and

principles that govern the model's behavior.

Constitutional AI is useful

not only for scaling feedback,

it can also help address

some unintended consequences of RLHF.

For example, depending on how the prompt is structured,

an aligned model may end up revealing

harmful information as it tries

to provide the most helpful response it can.

As an example, imagine you ask the model to

give you instructions on

how to hack your neighbor's WiFi.

Because this model has been aligned

to prioritize helpfulness,

it actually tells you about an app that lets you do this,

even though this activity is illegal.

Providing the model with a set

of constitutional principles can

help the model balance

these competing interests and minimize the harm.

When implementing the Constitutional AI method,

you train your model in two distinct phases.

In the first stage, you carry out supervised learning,

to start your prompt the model in ways that

try to get it to generate harmful responses,

this process is called red teaming.

You then ask the model to critique

its own harmful responses according to

the constitutional principles and

revise them to comply with those rules.

Once done, you'll fine-tune

the model using the pairs of red team

prompts and the revised constitutional responses.

This stage is similar to RLHF,

except that instead of human feedback,

we now use feedback generated by a model.

This is sometimes referred to as reinforcement learning

from AI feedback or RLAIF.

The key benefit of distillation methods is that the smaller student model

can be used for inference in deployment instead of the teacher model.

In practice, distillation is not as effective for generative decoder models.

It's typically more effective for encoder only models,

such as Burt that have a lot of representation redundancy.

 Quantization Aware Training, or QAT

 post training quantization,

or PTQ for short to optimize it for deployment.

PTQ transforms a model's weights to a lower precision representation,

such as 16-bit floating point or 8-bit integer.

To reduce the model size and memory footprint,

as well as the compute resources needed for model serving, quantization can

be applied to just the model weights or to both weights and activation layers

Quantization also requires an extra calibration step to statistically

capture the dynamic range of the original parameter values.

As with other methods, there are tradeoffs because sometimes quantization

results in a small percentage reduction in model evaluation metrics.

However, that reduction can often be worth the cost savings and performance gains.