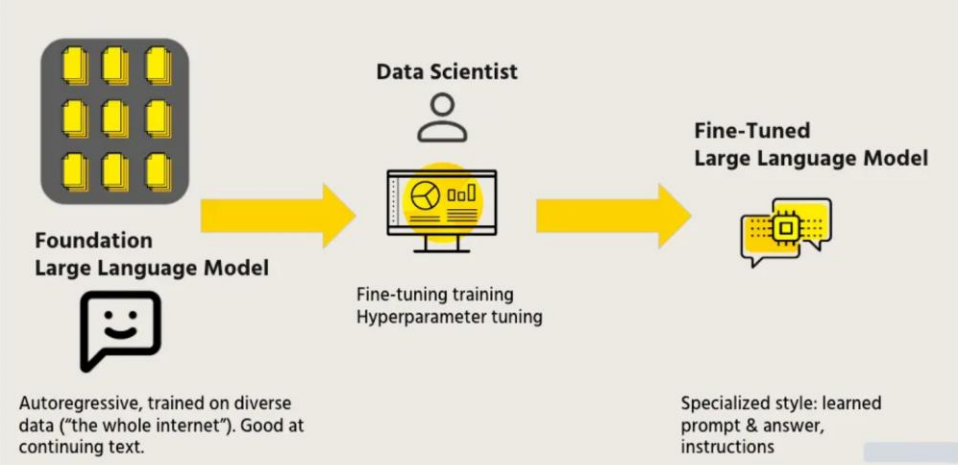
As described in my previous post, building LLM from scratch is a costly, time consuming and high resource intensive task. So, let’s discuss how one can use an open-source inference language model to solve a specific generative ai problem.

Finetuning LLMs

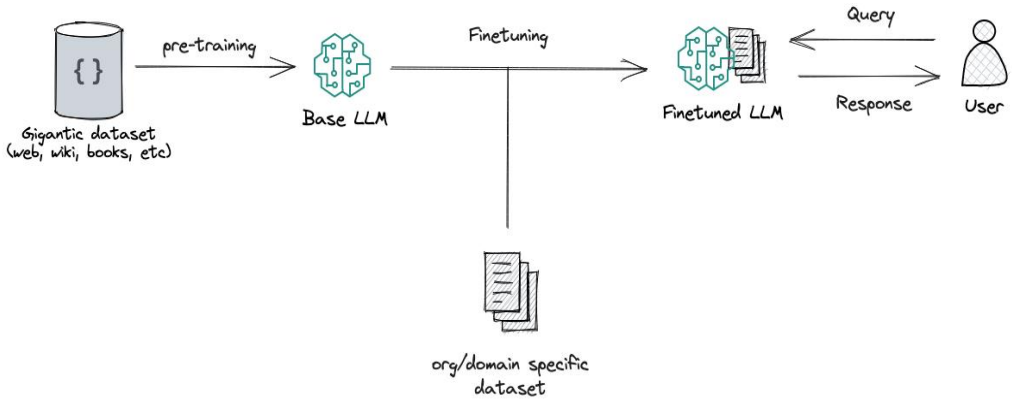


What is finetuning?

Fine-tuning is a process of transfer learning where we adjust the weights and parameters of a pre-trained model on new focused data to improve its performance for a specific task or industry use case. It involves training the model on a new dataset that is specific to the task at the same time updating the model's weights to adapt to the new data.

As demonstrated by OpenAI a small finetuned language model outperforms large base models on specific tasks.

The process of fine-tuning an LLM involves several key steps, including defining the project's vision and scope, selecting the right dataset, crafting effective prompts, evaluating the model's performance regularly, and deploying the model once it performs as expected.



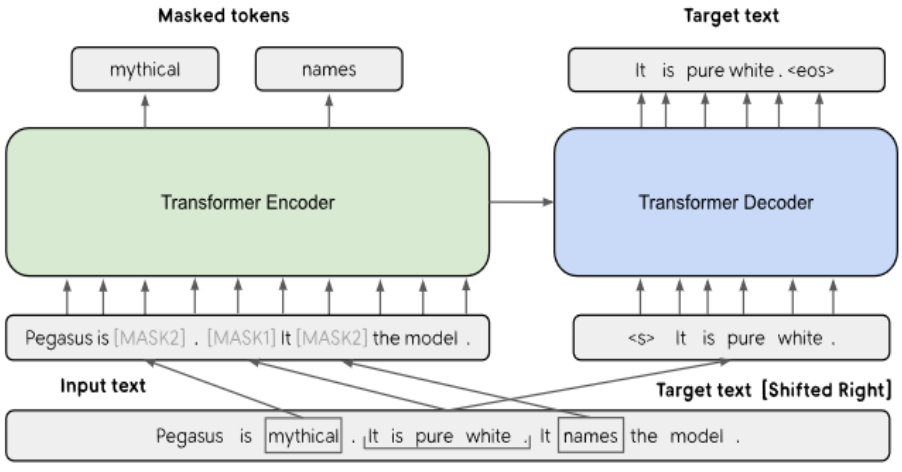
3 ways of fine-tuning

There are 3 ways we can opt to fine-tune a model

1. Self-supervised
2. Supervised
3. Reinforcement Learning

1. Self-supervised fine-tuning

Self-supervised learning involves training a model on pretext task where the target is derived from the input data itself. For LLMs, this often means predicting the next word when a group of words are given.



Steps to fine-tune a pre-trained LLM using self-supervised learning:

a. Define project's vision and scope

- Make a clear understanding of the specific task or problem you want to solve in the domain or industry.

b. Data Preparation:

- Collect and preprocess a domain-specific dataset.

- Mask words or create suitable pretext tasks for the self-supervised learning approach.

c. Model Architecture:

- Utilize the pre-trained LLM as the base architecture.

d. Hyperparameter Tuning/training:

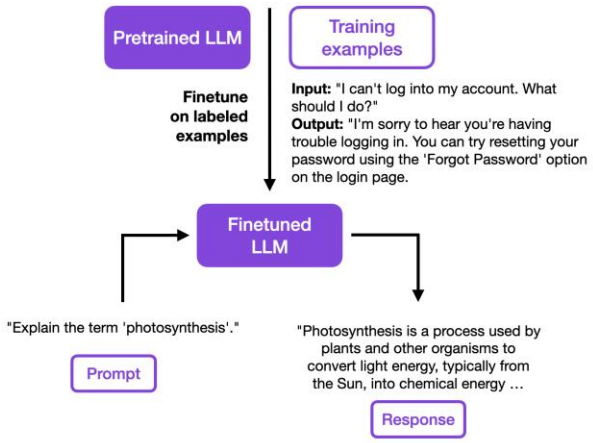
- Adjust learning rates, number of epochs, batch sizes, and other hyperparameters for optimal performance on the fine-tuning task.

e. Evaluate and refine:

- Test the fine-tuned model on unseen data to measure its performance and make further adjustments if necessary.

2. Supervised fine-tuning

In supervised fine-tuning, the model is trained on a labeled dataset specific to the desired task.



The steps include:

a. Define project's vision and scope
- Make a clear understanding of the specific task or problem you want to solve in the domain or industry.

b. Data preparation or Annotation:

- Annotate a dataset with labeled examples for the target task.

c. Model Configuration:

- Use the pre-trained LLM as the base model.

- Add task-specific output layers and configure them for the supervised learning task.

d. Fine-Tuning Process:

- Train the model on the annotated dataset using the labeled examples.

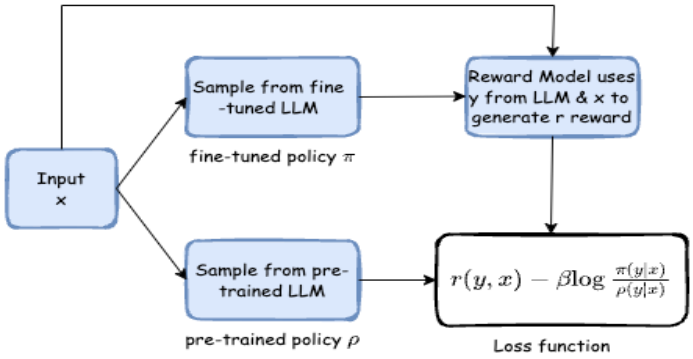
- Monitor performance metrics and adjust hyperparameters as needed.

e. Evaluate the model's performance

- Evaluate the performance of the model using new unseen data and update the hyper parameters to enhance the performance.

3. Reinforcement learning fine-tuning

Reinforcement learning fine-tuning involves training a model to make sequential decisions through interactions with an environment.



The steps include:

a. Define project's vision and scope
- Make a clear understanding of the specific task or problem you want to solve in the domain or industry.

b. Environment Setup:

- Define the task-specific environment and rewards for reinforcement learning.

c. Model Modification:

- Adjust the pre-trained LLM to handle sequential decision-making and reinforcement learning tasks.

d. Training Process:

- Iterate through episodes of interaction, adjusting the model parameters to maximize the cumulative reward.

e. Evaluation

We can now review the test results by providing new unknown data to the model.

Challenges in Fine-Tuning LLMs:

1. **Computational cost**:

It is impossible to fine-tune LLMs on consumer hardware, hence VRAMs are highly required for the fine-tuning process. So, we can use different cloud platforms like AWS, GCP, Azure to avail the service of GPU instances and we also can use free notebooks from google colab and kaggle. Using these platforms, we can overcome memory issues and other computing challenges. Or we can use parameter-efficient fine-tuning techniques like LoRA or QLoRA to optimize resource usage.

2. **Data Bias**:

Fine-tuning may exacerbate biases present in the pre-trained model or introduce new biases based on the fine-tuning dataset.

3. **Catastrophic Forgetting**:

The fine-tuning process may cause the model to forget information from the pre-training phase, impacting its generalization capabilities.

4. **Hyperparameter Sensitivity**:

Finding the right set of hyperparameters for fine-tuning can be challenging and may require extensive experimentation.

5. **Limited Task-Specific Data**:

In some cases, acquiring enough task-specific labeled data for supervised fine-tuning can be a bottleneck. Because obtaining high-quality, labeled data can be expensive and time-consuming. Techniques like transfer learning and data augmentation can help mitigate this.

Conclusion:

Fine-tuning LLMs opens a world of possibilities, allowing us to unlock the full potential of these powerful language models. By understanding the different approaches, steps involved, and potential challenges, we can harness this technology to create innovative solutions across various domains.

I hope this post has given you a good overview of fine-tuning LLMs! Feel free to ask any further questions you might have.

Stay tuned for the next post where I'll be discussing the technical side of it.

Connect with me: https://github.com/saisubhasish