[Week 3] Fine Tuning LLMs

ETMI5: Explain to Me in 5

In this section, we will go over the Fine-Tuning domain adaptation method for LLMs. Fine-tuning involves further training pre-trained models for specific tasks or domains, adapting them to new data distributions, and enhancing efficiency by leveraging pre-existing knowledge. It is crucial for tasks where generic models may not excel. Two main types of fine-tuning include unsupervised (updating models without modifying behavior) and supervised (updating models with labeled data). We emphasize on the popular supervised method-Instruction fine-tuning which augments input-output examples with explicit instructions for better generalization. We'll dig deeper into Reinforcement Learning from Human Feedback (RLHF) which incorporates human feedback for model fine-tuning and Direct Preference Optimization (DPO) that directly optimizes models based on user preferences. We provide an overview of Parameter-Efficient Fine-Tuning (PEFT) approaches as well where selective updates are made to model parameters, addressing computational challenges, memory efficiency, and allowing versatility across modalities.

Introduction

Fine-tuning is the process of taking pre-trained models and further training them on smaller, domain-specific datasets. The aim is to refine their capabilities and enhance performance in a specific task or domain. This process transforms general-purpose models into specialized ones, bridging the gap between generic pre-trained models and the unique requirements of particular applications.

Consider OpenAI's GPT-3, a state-of-the-art LLM designed for a broad range of NLP tasks. To illustrate the need for fine-tuning, imagine a healthcare organization wanting to use GPT-3 to assist doctors in generating patient reports from textual notes. While GPT-3 is proficient in general text understanding, it may not be optimized for intricate medical terms and specific healthcare jargon.

In this scenario, the organization engages in fine-tuning GPT-3 on a dataset filled with medical reports and patient notes. The model becomes more familiar with medical terminologies, nuances of clinical language, and typical report structures. As a result, after fine-tuning, GPT-3 is better suited to assist doctors in generating accurate and coherent patient reports, showcasing its adaptability for specific tasks.

Fine-tuning is not exclusive to language models; any machine learning model may require retraining under certain circumstances. It involves adjusting model parameters to align with the distribution of new, specific datasets. This process is illustrated with the example of a convolutional neural network trained to identify images of automobiles and the challenges it faces when applied to detecting trucks on highways.

The key principle behind fine-tuning is to leverage pre-trained models and recalibrate their parameters using novel data, adapting them to new contexts or applications. It is

particularly beneficial when the distribution of training data significantly differs from the requirements of a specific application.The choice of the base general model model depends on the nature of the task, such as text generation or text classification.

Why Fine-Tuning?

While large language models are indeed trained on a diverse set of tasks, the need for fine-tuning arises because these large generic models are designed to perform reasonably well across various applications, but not necessarily excel in a specific task. The optimization of generic models is aimed at achieving decent performance across a range of tasks, making them versatile but not specialized.

Fine-tuning becomes essential to ensure that a model attains exceptional proficiency in a particular task or domain of interest. The emphasis shifts from achieving general competence to achieving mastery in a specific application. This is particularly crucial when the model is intended for a focused use case, and overall general performance is not the primary concern.

In essence, generic large language models can be considered as being proficient in multiple tasks but not reaching the level of mastery in any. Fine-tuned models, on the other hand, undergo a tailored optimization process to become masters of a specific task or domain. Therefore, the decision to fine-tune models is driven by the necessity to achieve superior performance in targeted applications, making them highly effective specialists in their designated areas.

For a deeper understanding, explore why fine-tuning models for tasks in new domains is deemed crucial for several compelling reasons.

1. **Domain-Specific Adaptation:** Pre-trained LLMs may not be optimized for specific tasks or domains. Fine-tuning allows adaptation to the nuances and characteristics of a new domain, enhancing performance in domain-specific tasks. For instance, large generic LLMs might not be sufficiently trained on tasks like document analysis in the legal domain. Fine-tuning can allow the model to understand legal terminology and nuances for tasks like contract review.
2. **Shifts in Data Distribution:** Models trained on one dataset may not generalize well to out-of-distribution examples. Fine-tuning helps align the model with the distribution of new data, addressing shifts in data characteristics and improving performance on specific tasks. For example: Fine-tuning a sentiment analysis model for social media comments. The distribution of language and sentiments on social media may differ significantly from the original training data, requiring adaptation for accurate sentiment classification.
3. **Cost and Resource Efficiency:** Training a model from scratch on a new task often requires a large labeled dataset, which can be costly and time-consuming. Fine-tuning allows leveraging a pre-trained model's knowledge and adapting it to the new task with a smaller dataset, making the process more efficient. For example: Adapting a pre-trained model for a small e-commerce platform to recommend

products based on user preferences. Fine-tuning is more resource-efficient than training a model from scratch with a limited dataset.

4. **Out-of-Distribution Data Handling:**
   - Fine-tuning mitigates the suboptimal performance of pre-trained models when dealing with out-of-distribution examples. Instead of starting training anew, fine-tuning allows building upon the existing model's foundation with a relatively modest dataset. For example: Fine-tuning a speech recognition model for a new regional accent. The model can be adapted to recognize speech patterns specific to the new accent without extensive retraining.
5. **Knowledge Transfer:**
   - Pre-trained models capture general patterns and knowledge from vast amounts of data during pre-training. Fine-tuning facilitates the transfer of this general knowledge to specific tasks, making it a valuable tool for leveraging pre-existing knowledge in new applications. For example: Transferring medical knowledge from a pre-trained model to a new healthcare chatbot. Fine-tuning with medical literature enables the model to provide accurate and contextually relevant responses in healthcare conversations.
6. **Task-Specific Optimization:**
   - Fine-tuning enables the optimization of model parameters for task-specific objectives. For example, in the medical domain, fine-tuning an LLM with medical literature can enhance its performance in medical applications. For example: Optimizing a pre-trained model for code generation in a software development environment. Fine-tuning with code examples allows the model to better understand and generate code snippets.
7. **Adaptation to User Preferences:** Fine-tuning allows adapting the model to user preferences and specific task requirements. It enables the model to generate more contextually relevant and task-specific responses. For example: Fine-tuning a virtual assistant model to align with user preferences in language and tone. This ensures that the assistant generates responses that match the user's communication style.
8. **Continual Learning:** Fine-tuning supports continual learning by allowing models to adapt to evolving data and user requirements over time. It enables models to stay relevant and effective in dynamic environments. For instance: Continually updating a news summarization model to adapt to evolving news topics and user preferences. Fine-tuning enables the model to stay relevant and provide timely summaries.

In summary, fine-tuning is a powerful technique that enables organizations to adapt pre-trained models to specific tasks, domains, and user requirements, providing a practical and efficient solution for deploying models in real-world applications.

Types of Fine-Tuning

At a high level, fine-tuning methods for language models can be categorized into two main approaches: supervised and unsupervised. In machine learning, supervised methods involve having labeled data, where the model is trained on examples with corresponding

desired outputs. On the other hand, unsupervised methods operate with unlabeled data, focusing on extracting patterns and structures without explicit labels.

Unsupervised Fine-Tuning Methods:

1. **Unsupervised Full Fine-Tuning:** Unsupervised fine-tuning becomes relevant when there is a need to update the knowledge base of an LLM without modifying its existing behavior. For instance, if the goal is to fine-tune the model on legal literature or adapt it to a new language, an unstructured dataset containing legal documents or texts in the desired language can be utilized. In such cases, the unstructured dataset comprises articles, legal papers, or relevant content from authoritative sources in the legal domain. This approach allows the model to effectively refine its understanding and adapt to the nuances of legal language without relying on labeled examples, showcasing the versatility of unsupervised fine-tuning across various domains.
2. **Contrastive Learning:** Contrastive learning is a method employed in fine-tuning language models, emphasizing the training of the model to discern between similar and dissimilar examples in the latent space. The objective is to optimize the model's ability to distinguish subtle nuances and patterns within the data. This is achieved by encouraging the model to bring similar examples closer together in the latent space while pushing dissimilar examples apart. The resulting learned representations enable the model to capture intricate relationships and differences in the input data. Contrastive learning is particularly beneficial in tasks where a nuanced understanding of similarities and distinctions is crucial, making it a valuable technique for refining language models for specific applications that require fine-grained discrimination.

Supervised Fine-Tuning Methods:

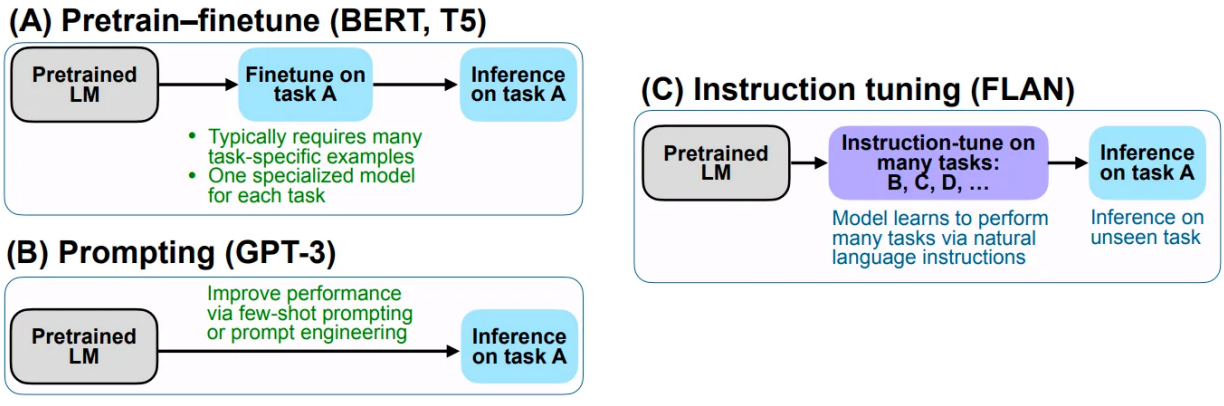
1. **Parameter-Efficient Fine-Tuning: It** is a fine-tuning strategy that aims to reduce the computational expenses associated with updating the parameters of a language model. Instead of updating all parameters during fine-tuning, PEFT focuses on selectively updating a small set of parameters, often referred to as a low-dimensional matrix. One prominent example of PEFT is the low-rank adaptation (LoRA) technique. LoRA operates on the premise that fine-tuning a foundational model for downstream tasks only requires updates across certain parameters. The low-rank matrix effectively represents the relevant space related to the target task, and training this matrix is performed instead of adjusting the entire model's parameters. PEFT, and specifically techniques like LoRA, can significantly decrease the costs associated with fine-tuning, making it a more efficient process.
2. **Supervised Full Fine-Tuning**: It involves updating all parameters of the language model during the training process. Unlike PEFT, where only a subset of parameters is modified, full fine-tuning requires sufficient memory and computational resources to store and process all components being updated. This comprehensive approach results in a new version of the model with updated weights across all layers. While full fine-tuning is more resource-intensive, it ensures that the entire model is adapted to the specific task or domain, making it suitable for situations where a thorough adjustment of the language model is desired.

3. **Instruction Fine-Tuning:** Instruction Fine-Tuning involves the process of training a language model using examples that explicitly demonstrate how it should respond to specific queries or tasks. This method aims to enhance the model's performance on targeted tasks by providing explicit instructions within the training data. For instance, if the task involves summarization or translation, the dataset is curated to include examples with clear instructions like "summarize this text" or "translate this phrase." Instruction fine-tuning ensures that the model becomes adept at understanding and executing specific instructions, making it suitable for applications where precise task execution is essential.
4. **Reinforcement Learning from Human Feedback (RLHF):** RLHF takes the concept of supervised fine-tuning a step further by incorporating reinforcement learning principles. In RLHF, human evaluators are enlisted to rate the model's outputs based on specific prompts. These ratings serve as a form of reward, guiding the model to optimize its parameters to maximize positive feedback. RLHF is a resource-intensive process that leverages human preferences to refine the model's behavior. Human feedback contributes to training a reward model that guides the subsequent reinforcement learning phase, resulting in improved model performance aligned with human preferences.

Techniques such as contrastive learning, as well as supervised and unsupervised fine-tuning, are not exclusive to LLMs and have been employed for domain adaptation even before the advent of LLMs. However, following the rise of LLMs, there has been a notable increase in the prominence of techniques such as RLHF, instruction fine-tuning, and PEFT. In the upcoming sections, we will explore these methodologies in greater detail to comprehend their applications and significance.

Instruction Fine-Tuning

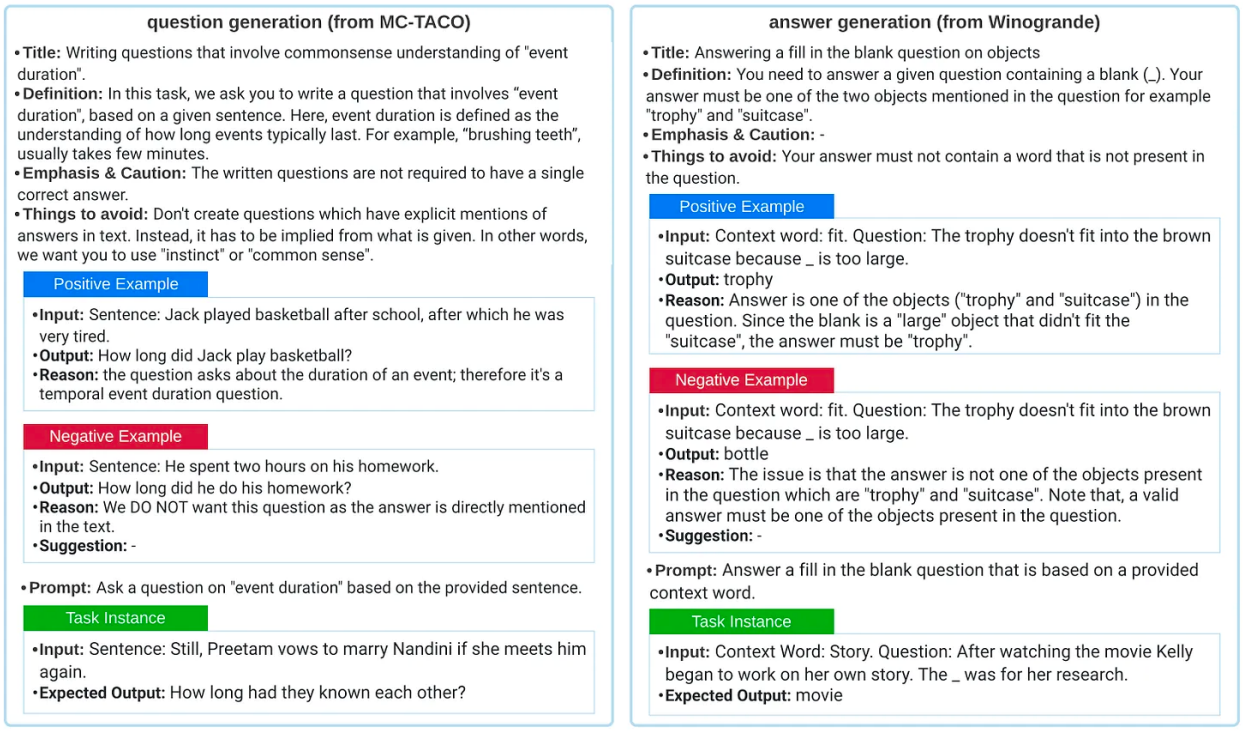
Instruction fine-tuning is a method that has gained prominence in making LLMs more practical for real-world applications. In contrast to standard supervised fine-tuning, where models are trained on input examples and corresponding outputs, instruction tuning involves augmenting input-output examples with explicit instructions. This unique approach enables instruction-tuned models to generalize more effectively to new tasks. The data for instruction tuning is constructed differently, with instructions providing additional context for the model.



finetuning.png

Image Source: Wei et al., 2022

One notable dataset for instruction tuning is “Natural Instructions”. This dataset consists of 193,000 instruction-output examples sourced from 61 existing English NLP tasks. The uniqueness of this dataset lies in its structured approach, where crowd-sourced instructions from each task are aligned to a common schema. Each instruction is associated with a task, providing explicit guidance on how the model should respond. The instructions cover various fields, including a definition, things to avoid, and positive and negative examples. This structured nature makes the dataset valuable for fine-tuning models, as it provides clear and detailed instructions for the desired task. However, it’s worth noting that the outputs in this dataset are relatively short, which might make the data less suitable for generating long-form content. Despite this limitation, Natural Instructions serves as a rich resource for training models through instruction tuning, enhancing their adaptability to specific NLP tasks. The below image contains an example instruction format



finetuning_1.png

Image Source: **Mishra et al., 2022**

Instruction fine-tuning has become a valuable tool in the evolving landscape of natural language processing and machine learning, enabling LLMs to adapt to specific tasks with nuanced instructions.

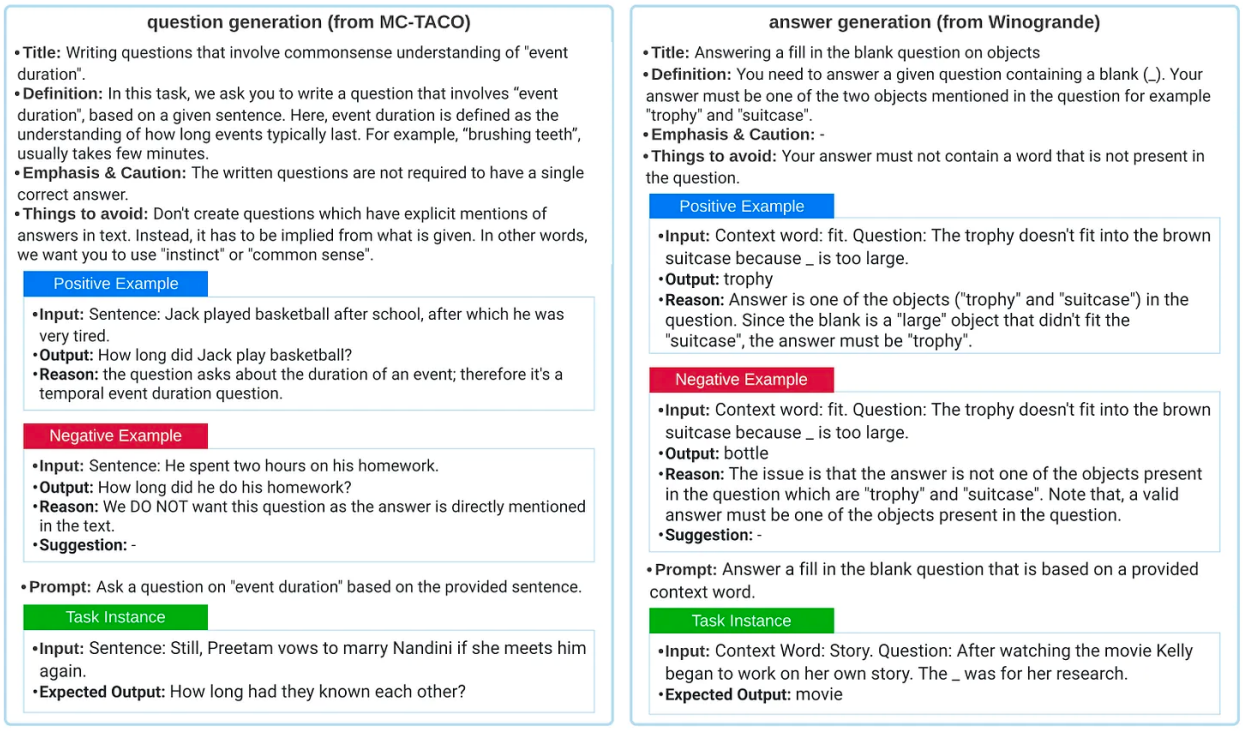
Reinforcement Learning from Human Feedback (RLHF)

Reinforcement Learning from Human Feedback is a methodology designed to enhance language models by incorporating human feedback, aligning them more closely with intricate human values. The RLHF process comprises three fundamental steps:

1. Pretraining Language Models (LMs): RLHF initiates with a pretrained LM, typically achieved through classical pretraining objectives. The initial LM, which can vary in size, is flexible in choice. While optional, the initial LM can undergo fine-tuning on additional data. The crucial aspect is to have a model that exhibits a positive response to diverse instructions.

2. Reward Model Training: The subsequent step involves generating a reward model (RM) calibrated with human preferences. This model assigns scalar rewards to sequences of text, reflecting human preferences. The dataset for training the reward model is generated by sampling prompts and passing them through the initial LM to produce text. Human annotators rank the generated text outputs, and these rankings are used to create a regularized dataset for training the reward model. The reward function combines the preference model and a penalty on the difference between the RL policy and the initial model.

3. Fine-Tuning with RL: The final step entails fine-tuning the initial LLM using reinforcement learning. Proximal Policy Optimization (PPO) is a commonly used RL algorithm for this task. The RL policy is the LM that takes in a prompt and produces text, with actions corresponding to tokens in the LM's vocabulary. The reward function, derived from the preference model and a constraint on policy shift, guides the fine-tuning. PPO updates the LM's parameters to maximize the reward metrics in the current batch of prompt-generation pairs. Some parameters of the LM are frozen due to computational constraints, and the fine-tuning aims to align the model with human preferences.



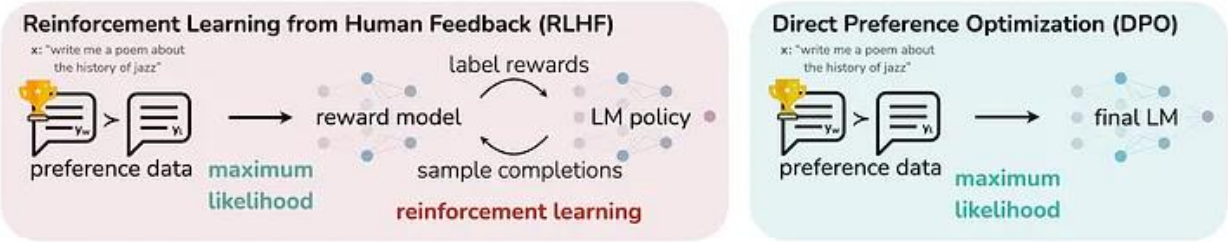
finetuning_2.png

Image Source: https://openai.com/research/instruction-following

💡 If you're lost understanding RL terms like PPO, policy etc. Think of this analogy- Fine-Tuning with RL, specifically using Proximal Policy Optimization (PPO), is similar to refining instructions to train a pet, such as teaching a dog tricks. Think of the dog initially learning with general guidance (policy) and receiving treats (rewards) for correct actions. Now, imagine the dog mastering a new trick but not quite perfectly. Fine-tuning, with PPO, involves adjusting your instructions slightly based on how well the dog performs, similar to tweaking the model's behavior (policy) in Reinforcement Learning. It's like refining the instructions to optimize the learning process, much like perfecting your pet's tricks through gradual adjustments and treats for better performance.

Direct Preference Optimization DPO (*Bonus Topic)*

Direct Preference Optimization (DPO) is an equivalent of RLHF and has been gaining significant traction these days. DPO offers a straightforward method for fine-tuning large language models based on human preferences. It eliminates the need for a complex reward model and directly incorporates user feedback into the optimization process. In DPO, users simply compare two model-generated outputs and express their preferences, allowing the LLM to adjust its behavior accordingly. This user-friendly approach comes with several advantages, including ease of implementation, computational efficiency, and greater control over the LLM's behavior.



finetuning_3.png

Image Source: Rafailov, Rafael, et al.

💡 In the context of LLMs, maximum likelihood is a principle used during the training of the model. Imagine the model is like a writer trying to predict the next word in a sentence. Maximum likelihood training involves adjusting the model's parameters (the factors that influence its predictions) to maximize the likelihood of generating the actual sequences of words observed in the training data. It's like tuning the writer's skills to make the sentences they create most closely resemble the sentences they've seen before. So, maximum likelihood helps the LLM learn to generate text that is most similar to the examples it was trained on.

DPO*:* DPO takes a straightforward approach by directly optimizing the LM based on user preferences without the need for a separate reward model. Users compare two model-generated outputs, expressing their preferences to guide the optimization process.

RLHF*:* RLHF follows a more structured path, leveraging reinforcement learning principles. It involves training a reward model that learns to identify and reward desirable LM outputs. The reward model then guides the LM's training process, shaping its behavior towards achieving positive outcomes.

DPO (Direct Policy Optimization) vs. RLHF (Reinforcement Learning from Human Feedback): Understanding the Differences

DPO - A Simpler Approach: Direct Policy Optimization (DPO) takes a straightforward path, sidestepping the need for a complex reward model. It directly optimizes the Large Language Model (LLM) based on user preferences, where users compare two outputs and indicate their preference. This simplicity results in key advantages:

1. **Ease of Implementation:** DPO is more user-friendly as it eliminates the need for designing and training a separate reward model, making it accessible to a broader audience.
2. **Computational Efficiency:** Operating directly on the LLM, DPO leads to faster training times and lower computational costs compared to RLHF, which involves multiple phases.
3. **Greater Control:** Users have direct control over the LLM's behavior, guiding it toward specific goals and preferences without the complexities of RLHF.

4. **Faster Convergence:** Due to its simpler structure and direct optimization, DPO often achieves desired results faster, making it suitable for tasks with rapid iteration needs.
5. **Improved Performance:** Recent research suggests that DPO can outperform RLHF in scenarios like sentiment control and response quality, particularly in summarization and dialogue tasks.

RLHF - A More Structured Approach: It follows a more structured path, leveraging reinforcement learning principles. It includes three training phases: pre-training, reward model training, and fine-tuning with reinforcement learning. While flexible, RLHF comes with complexities:

1. **Complexity:** RLHF can be more complex and sometimes unstable, demanding more computational resources and dealing with challenges like convergence, drift, or uncorrelated distribution problems.
2. **Flexibility in Defining Rewards:** RLHF allows for more nuanced reward structures, beneficial for tasks requiring precise control over the LLM's output.
3. **Handling Diverse Feedback Formats:** RLHF can handle various forms of human feedback, including numerical ratings or textual corrections, whereas DPO primarily relies on binary preferences.
4. **Handling Large Datasets:** RLHF can be more efficient in handling massive datasets, especially with distributed training techniques.

In summary, the choice depends on the specific task, available resources, and the desired level of control, with both methods offering strengths and weaknesses in different contexts. As advancements continue, these methods contribute to evolving and enhancing fine-tuning processes for LLMs.

Parameter Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) addresses the resource-intensive nature of fine-tuning LLMs. Unlike full fine-tuning that modifies all parameters, PEFT fine-tunes only a small subset of additional parameters while keeping the majority of pretrained model weights frozen. This selective approach minimizes computational requirements, mitigates catastrophic forgetting, and facilitates fine-tuning even with limited computational resources. PEFT, as a whole, offers a more efficient and practical method for adapting LLMs to specific downstream tasks without the need for extensive computational power and memory.

Advantages of Parameter-Efficient Fine-Tuning (PEFT)

1. **Computational Efficiency:** PEFT fine-tunes LLMs with significantly fewer parameters than full fine-tuning. This reduces the computational demands, making it feasible to fine-tune on less powerful hardware or in resource-constrained environments.

2. **Memory Efficiency:** By freezing the majority of pretrained model weights, PEFT avoids excessive memory usage associated with modifying all parameters. This makes PEFT particularly suitable for tasks where memory constraints are a concern.
3. **Catastrophic Forgetting Mitigation:** PEFT prevents catastrophic forgetting, a phenomenon observed during full fine-tuning where the model loses knowledge from its pre-trained state. This ensures that the LLM retains valuable information while adapting to new tasks.
4. **Versatility Across Modalities:** PEFT extends beyond natural language processing tasks, proving effective in various modalities such as computer vision and audio. Its versatility makes it applicable to a wide range of downstream tasks.
5. **Modular Adaptation for Multiple Tasks:** The modular nature of PEFT allows the same pretrained model to be adapted for multiple tasks by adding small task-specific weights. This avoids the need to store full copies for different applications, enhancing flexibility and efficiency.
6. **INT8 Tuning:** PEFT's capabilities include INT8 (8-bit integer) tuning, showcasing its adaptability to different quantization techniques. This enables fine-tuning even on platforms with limited computational resources.

In summary, PEFT offers a practical and efficient solution for fine-tuning large language models, addressing computational and memory challenges while maintaining performance on downstream tasks.

A summary of the most popular PEFT methods are in the chart below. Please download for improved visibility.

PEFT (1).pdf

Read/Watch These Resources (Optional)

1. https://www.superannotate.com/blog/llm-fine-tuning
2. https://www.deeplearning.ai/short-courses/finetuning-large-language-models/
3. https://www.youtube.com/watch?v=eC6Hd1hFvos
4. https://www.labellerr.com/blog/comprehensive-guide-for-fine-tuning-of-llms/

Read These Papers (Optional)

1. https://arxiv.org/abs/2303.15647
2. https://arxiv.org/abs/2109.10686
3. https://arxiv.org/abs/2304.01933