Advanced Generative AI: Building LLM Applications

LLM Fine-Tuning and Customization

Quick Recap

- How do embeddings in Generative AI contribute to LangChain's capabilities, and what role do document loaders play in the LangChain framework?
- How do chains connect models and enhance their capabilities within the LangChain framework?

Engage and Think

Imagine you have created a chatbot for the healthcare sector, but now you want to make it for XYZ Hospital. It's like introducing your chatbot to your hospital's datasets. You want it to understand the medical lingo specific to the place by training it with the hospital data. It's like customizing the chatbot's responses to handle the hospital's unique scenarios, making it more like a local expert.

How you can enhance the healthcare chatbot's functionality by training it with hospital-specific data, customizing responses for unique scenarios, and transforming it into a personalized virtual assistant for the hospital.

Learning Objectives

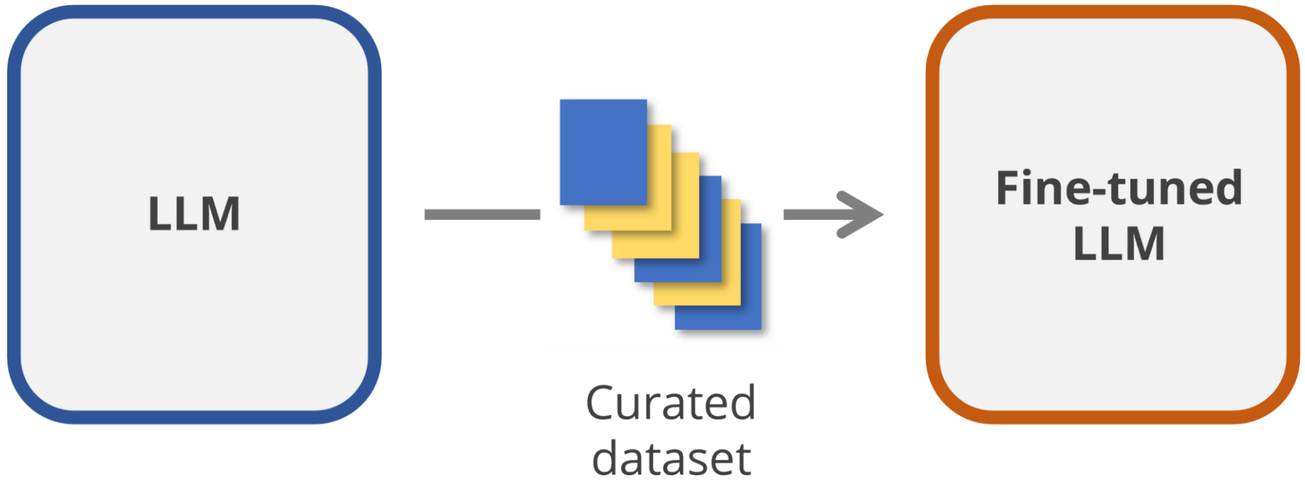
By the end of this lesson, you will be able to:

- Infer the concept, importance, and strategies of fine-tuning in LLMs for specific tasks
- Develop skills in data preparation and preprocessing for fine-tuning
- Outline insight into the intricacies of hyperparameter tuning processes for optimal performance
- Extend the concept of reinforcement learning with human feedback (RLHF), its applications, and its integration with fine-tuning

Introduction to LLM Fine-Tuning

Introduction to LLM Fine-Tuning

LLM fine-tuning involves refining a language model for specific tasks using a curated dataset.



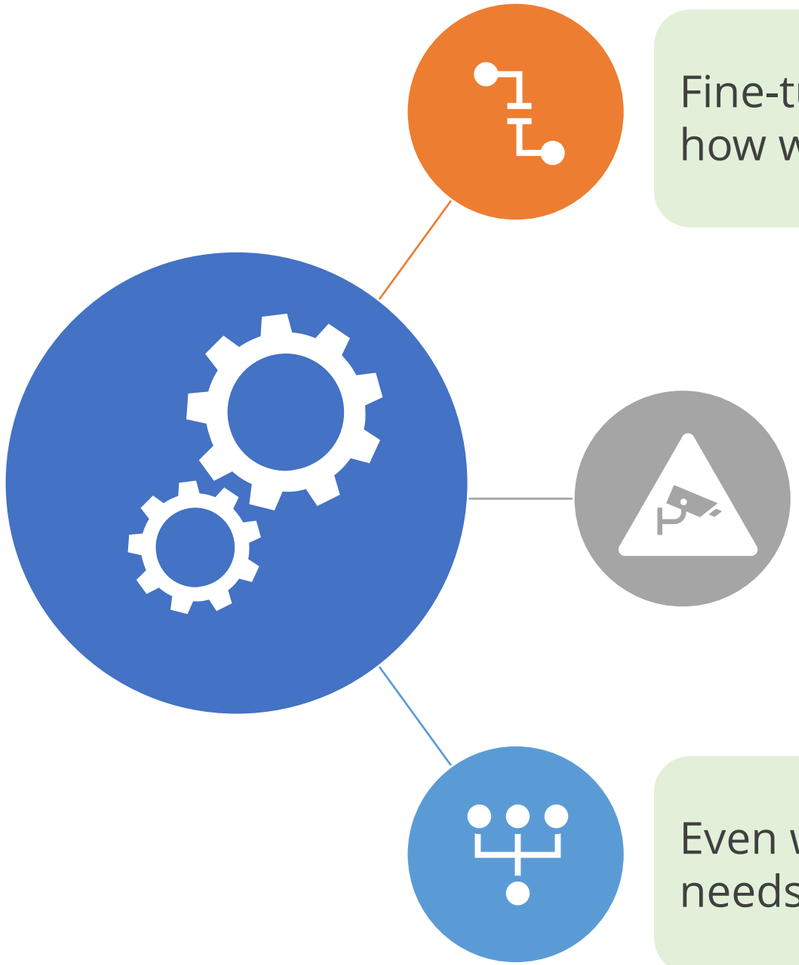
Fundamentals of Fine-Tuning of LLM

Fine-tuning LLM means customizing them for specific tasks using labeled data that fits the job.

Fine-tuning helps LLMs understand the connections between inputs and outputs in a specific dataset, refining their predictions.

The fine-tuning process can be time-consuming and requires a good dataset for fine-tuning, but it makes LLMs more efficient and effective for specialized tasks.

Need of Fine-Tuning



Fine-tuning LLMs is crucial to make them fit specific tasks, improving how well they work and relate to the context.

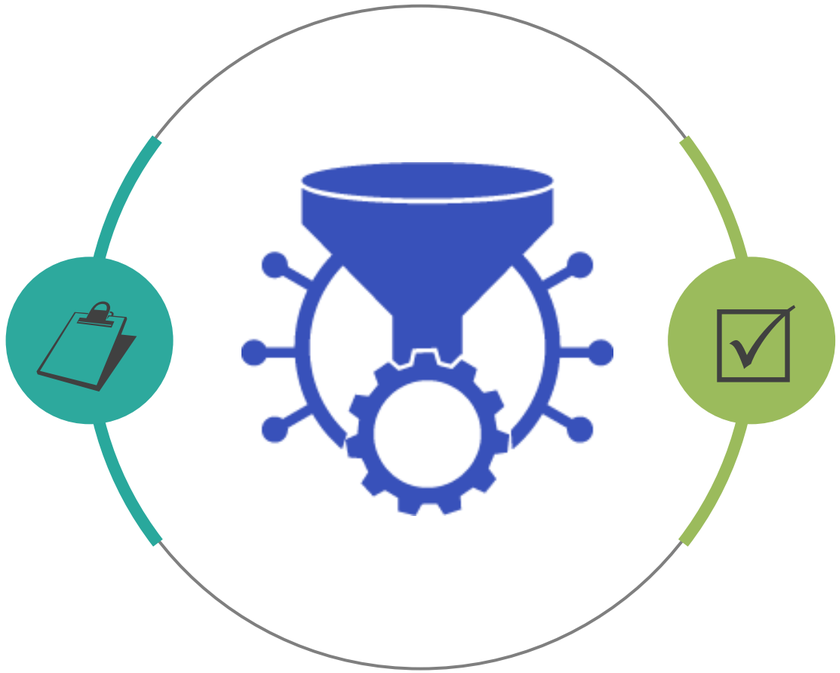
It's useful to make LLMs specific to a domain, handle rare cases, and keep data private and secure.

Even with challenges, fine-tuning is vital for tailoring LLMs to specific needs, especially for organizations using them in applications.

Data Preparation for LLM Tuning

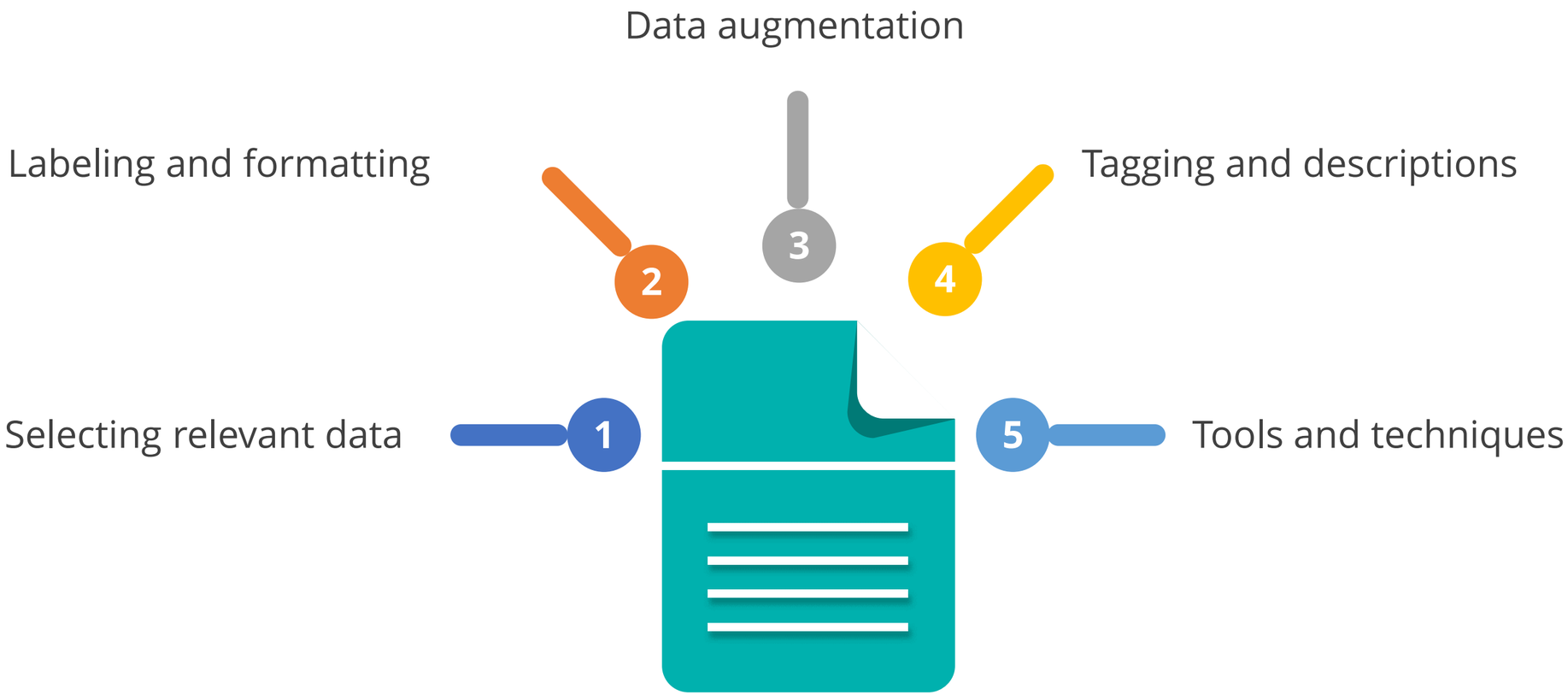
Data preparation is vital for fine-tuning LLMs to ensure the model excels in specific tasks.

The LLM fine-tuning process has two phases:



Data Preparation for LLM Tuning

Here are some key considerations and best practices for data preparation in LLM fine-tuning:



Demo: Data Preparation for Fine-Tuning

Duration: 10 minutes

In this demo, we will walk through the process of preparing data for fine-tuning a Language Learning Model (LLM). We will be using EleutherAI's Pythia-70m model, which is a transformer-based model designed for a variety of natural language processing tasks. The goal is to prepare a dataset of instructions for fine-tuning this model.

Note

Please download the solution document from the Reference Material Section and follow the Jupyter Notebook for step-by-step execution.

DEMONSTRATION

Quick Check

Which of the following is NOT a key consideration in the data preparation process for fine-tuning LLMs?

A. Labeling and formatting of the data

B. Utilizing data augmentation techniques

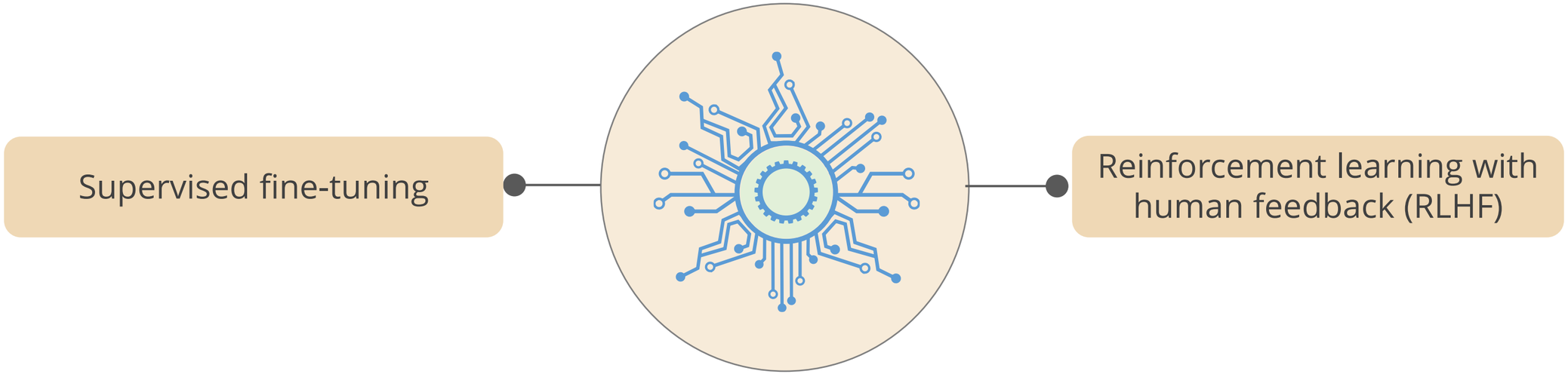
C. Selecting a dataset that is irrelevant to the specific task or domain

D. Using appropriate tools and techniques for data preparation to ensure data quality

Fine-Tuning Methodologies

Fine-Tuning Methodologies

Fine-tuning methodologies for LLMs can be broadly categorized into two major approaches:



Fine-Tuning Methodologies

Supervised fine-tuning

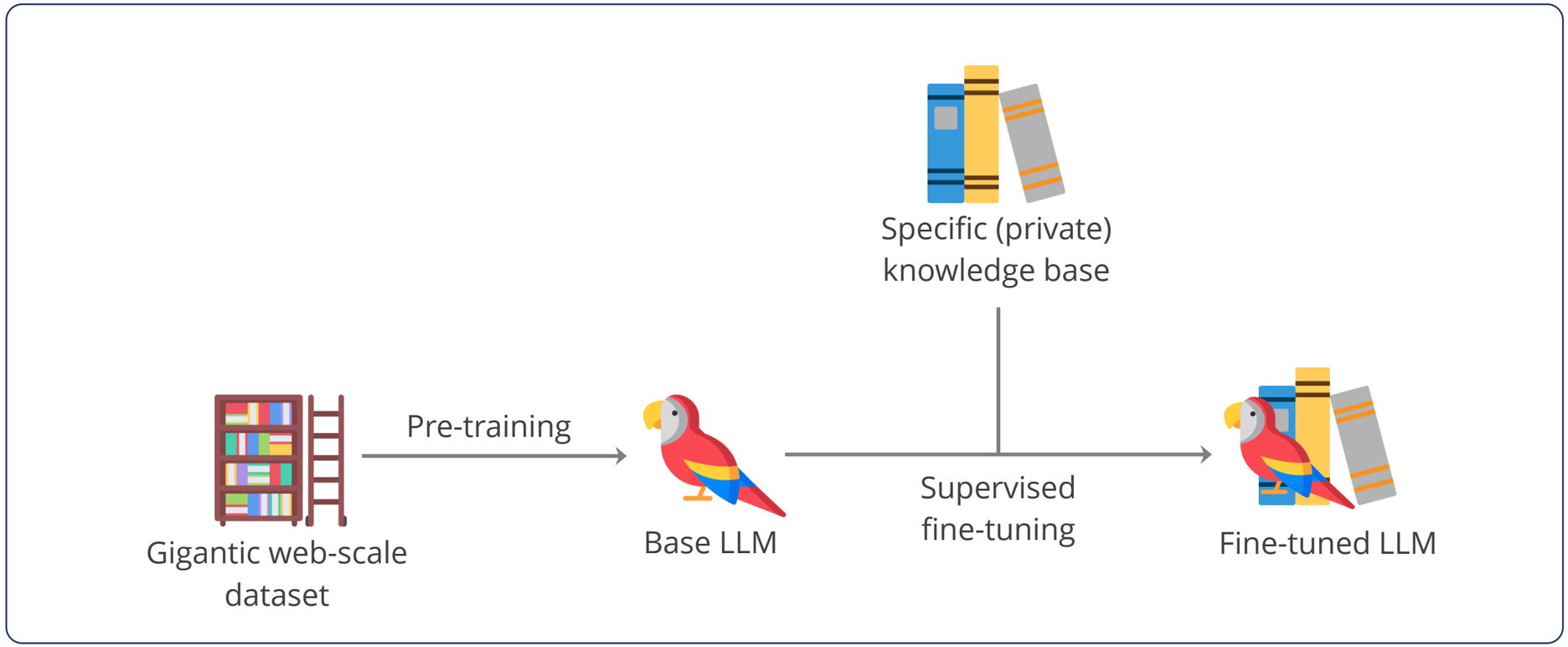
It customizes the model for a task with labeled data; the target model borrows designs from the source model, tweaking parameters except the output layer. This boosts transfer learning and accuracy.

Reinforcement learning with human feedback

The RLHF method aims for gradual improvements, trying strategies like regularization or changing the model structure. Engineers can refine the model iteratively until it reaches the desired performance level.

Supervised fine-tuning

Supervised fine-tuning is like giving a model extra training with specific examples and clear labels. It helps the model get a better at understanding and predicting things accurately.



Parameter-Efficient Fine-Tuning (PEFT)

PEFT is a technique used in NLP to improve the performance of pretrained language models on specific tasks while significantly reducing computational requirements.

PEFT enables fine-tuning a small subset of parameters in a pretrained LLM, making it more efficient and cost-effective.

Parameter-Efficient Fine-Tuning (PEFT)

Here are a few popular parameter-efficient fine-tuning (PEFT) methods:

LoRA (low-rank adapters)

It introduces trainable parameters into the transformer layers to enable the model to learn specific representations.

P-tuning (prompt tuning)

It adds a learnable prompt to the input to help the model adapt to new tasks.

Prefix tuning

It enhances task-specific learning in transformers by adding trainable tensors to each block.

Parameter-Efficient Fine-Tuning (PEFT)

Adapters

They add tunable layers to LLM's transformer blocks for task learning.

AdaLoRA

It extends LoRA, enabling fine-tuning on quantized weights for efficient training.

PEFT Techniques: LoRA

It adds low-rank trainable layers instead of modifying the entire model while keeping most of the model weights frozen. Key points include:

Reduces GPU memory usage – Fine-tunes LLMs without storing all gradients.

Efficient training – Needs only 0.1% – 1% of the model parameters.

Maintains model performance – Comparable results to full fine-tuning.

Ideal for domain-specific fine-tuning – Allows fine-tuning of large-scale models on limited hardware.

PEFT Techniques: LoRA Example

Here is an example of fine-tuning GPT-3 for legal document summarization:

LoRA

```
from peft import get_peft_model, LoraConfig, TaskType
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load pre-trained model and tokenizer
model_name = "meta-llama/Llama-2-7b"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Define LoRA configuration
lora_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    r=8, lora_alpha=32, lora_dropout=0.1
)

# Apply LoRA adaptation
lora_model = get_peft_model(model, lora_config)
lora_model.train()
```

PEFT Techniques: P-Tuning

P-Tuning trains special word-like embeddings instead of changing the model itself. Instead of writing prompts manually, it learns the best way to guide the model for better performance. Key points include:

No need to modify model weights – Keeps LLM frozen and learns soft prompts instead.

Highly parameter-efficient – Uses only 0.1% – 0.5% of model parameters.

Works well for instruction-based tasks like text classification, dialogue generation, and few-shot learning.

PEFT Techniques: P-Tuning Example

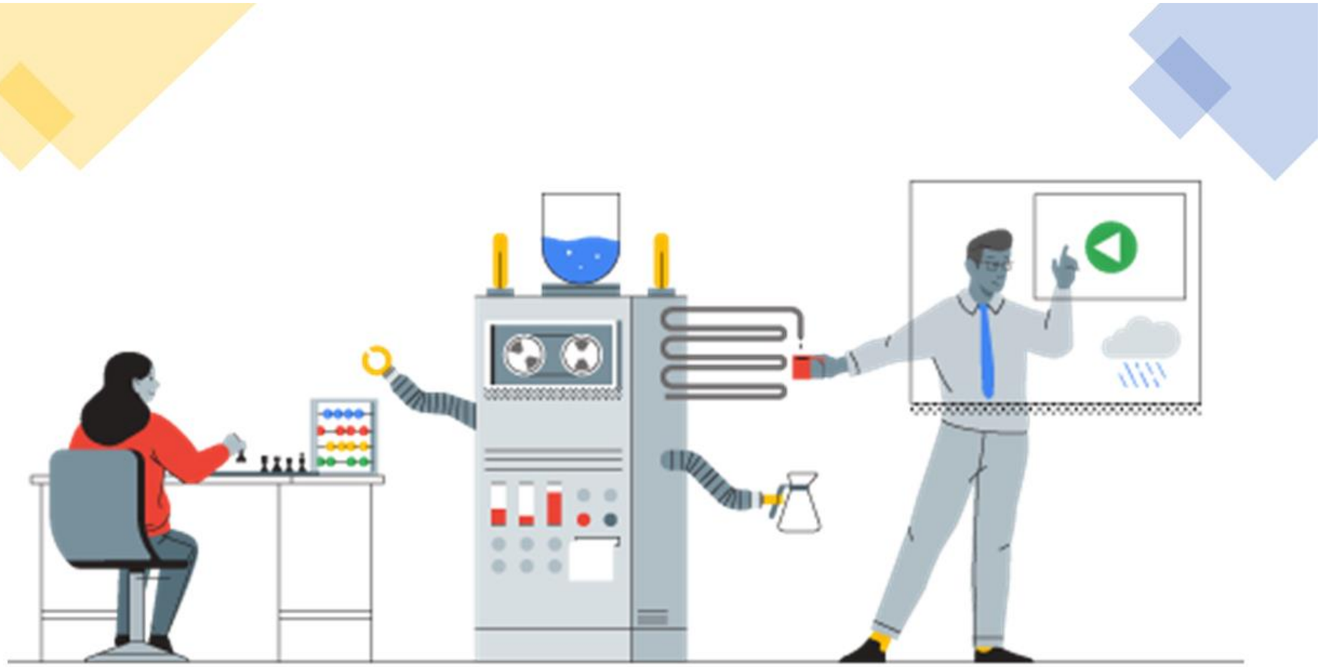
Here is an example of improving sentiment analysis with P-tuning:

P-tuning

```
from peft import get_peft_model, PromptTuningConfig, TaskType
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load pre-trained model and tokenizer
model_name = "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name, num_labels=2)

# Define P-Tuning configuration
p_tuning_config = PromptTuningConfig(
    task_type=TaskType.SEQ_CLS,
    num_virtual_tokens=20
)

# Apply P-Tuning
p_tuned_model = get_peft_model(model, p_tuning_config)
p_tuned_model.train()
```

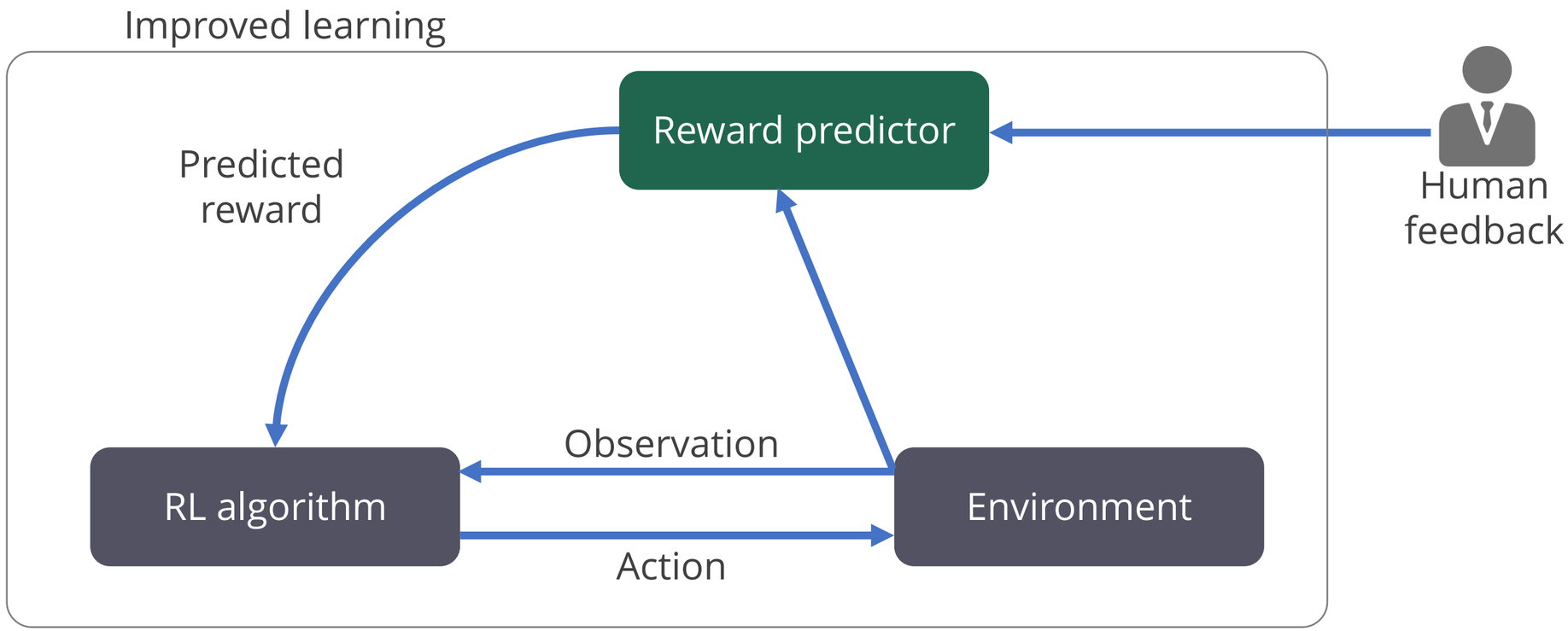
Reinforcement Learning with Human Feedback (RLHF)

RLHF is about training LLMs using human feedback, helping them follow instructions, and providing quality judgments.



Reinforcement Learning with Human Feedback (RLHF)

This phrase refers to a type of machine learning where a model is trained and adjusted based on feedback provided by humans.



Reinforcement Learning with Human Feedback (RLHF)

The key features of RLHF include the following:

RLHF refines LLMs by training them with human feedback, helping them follow instructions, and providing quality responses.

The RL policy, a copy of the original model, adjusts its behavior based on feedback to generate responses preferred according to rewards.

RLHF uses human feedback to train and fine-tune LLMs, improving their ability to provide more relevant and contextually fitting outputs. This enhances their performance in language applications.

RLHF: Example

Enhancing news summaries with human feedback

Problem

A news organization deploys an LLM to generate daily news summaries. Initially, the summaries may be too vague, omit critical details, or introduce biases.

RLHF Example: Human Feedback Process

Human feedback acts as a guiding force, refining AI-generated news summaries to be more accurate, clear, and comprehensive through iterative reinforcement.

RLHF Example: Outcomes

Through continuous learning from human feedback, the model evolves to produce summaries that are clearer, more accurate, and aligned with editorial standards.

Iterative improvement

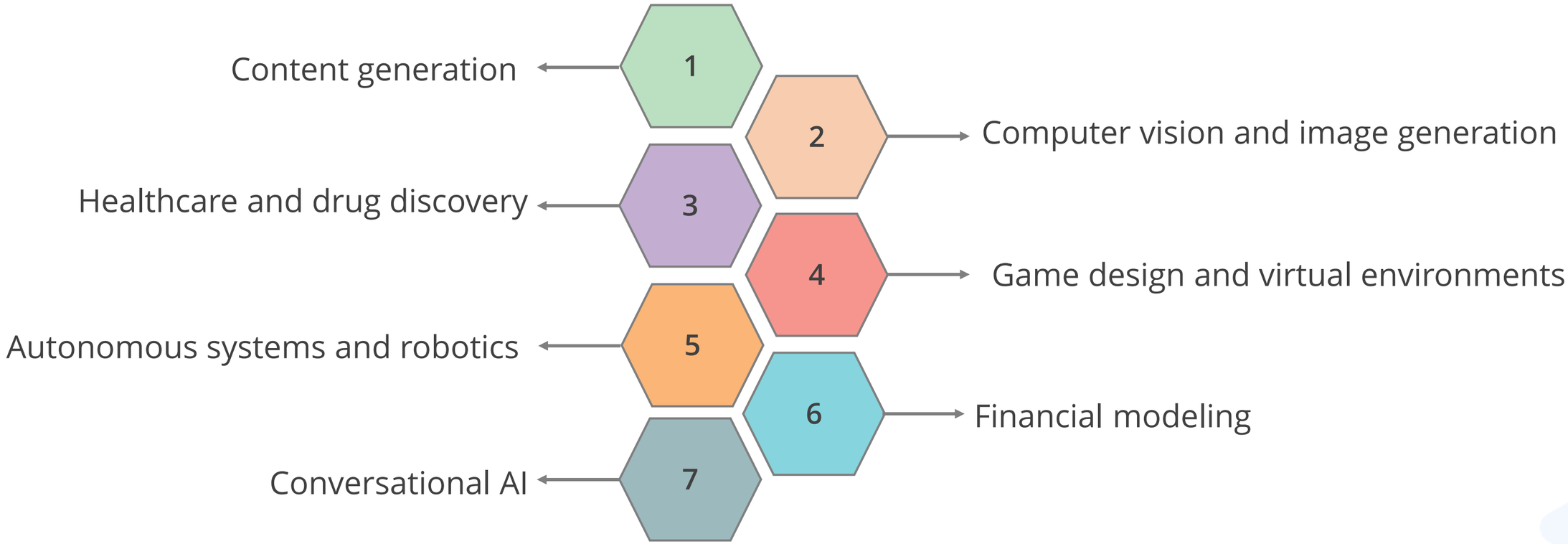
- Over multiple training cycles, the model learns to capture key details, improve clarity, and ensure factual accuracy.
- Biases are reduced as the model aligns its tone with journalistic standards.

Outcome

- Summaries become concise, well-structured, and factually robust.
- The model adapts to editorial preferences, ensuring consistency in news reporting.

Applications of RLHF

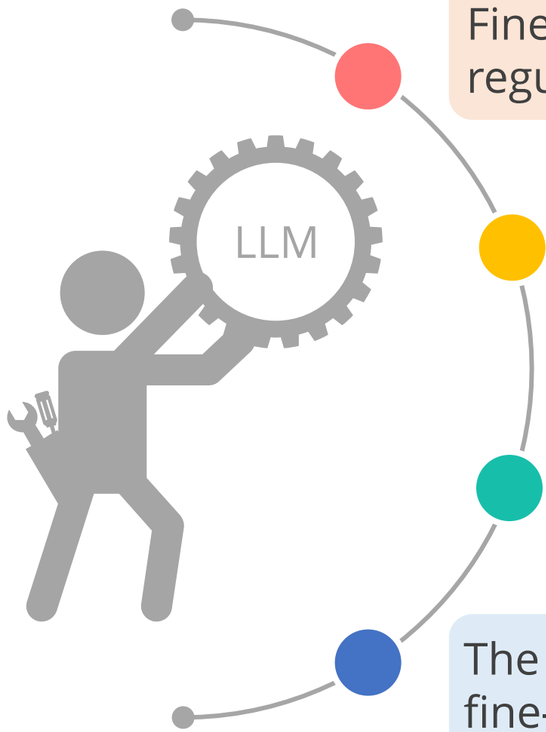
The various applications where reinforcement learning with human feedback is employed are:



Hyperparameter Tuning in Fine-Tuning

Hyperparameter adjusting is essential when fine-tuning language models for optimal performance. These parameters are set before the model learns, helping to shape its performance.

Hyperparameter Tuning



Fine-tuning includes adjusting key hyperparameters like learning rate, batch size, regularization strength, and number of epochs.

Tuning hyperparameters means finding the best combo for effective learning without sacrificing speed or risking overfitting.

Hyperparameter tuning methods vary, from basic manual adjustments to advanced approaches like Bayesian optimization and grid search.

The method chosen depends on the task, data, and resources. It is crucial for LLM fine-tuning, impacting performance.

Fine-Tuning vs. RAG

The following are key differences between fine-tuning, which embeds domain expertise into model parameters, and retrieval-augmented generation, which leverages dynamic external information.

Criteria	Fine-tuning	RAG
What they are	Adjusts a pre-trained model's parameters using domain-specific data to deeply embed specialized knowledge	Enhances model responses by retrieving and incorporating relevant external information during generation without altering model weights
Key advantages	- Deep integration of domain knowledge - Optimized performance for specialized tasks	- Rapid adaptation to evolving data - Lower computational overhead (no full retraining) - Ability to update content on the fly

Fine-Tuning vs. RAG

The following are key differences between fine-tuning, which embeds domain expertise into model parameters, and retrieval-augmented generation, which leverages dynamic external information.

Criteria	Fine-tuning	RAG
Considerations & trade-offs	- Requires significant computational resources and careful data curation - Risk of overfitting if not managed well	- Relies on the quality and relevance of the external corpus - May introduce retrieval latency and integration complexity
When to use which?	Ideal when you have a stable, well-defined dataset and need deep, domain-specific customization.	Best when your application benefits from up-to-date information and flexible, dynamic knowledge incorporation.

Quick Check

Which of the following PEFT methods involves adding a learnable prompt at the beginning of the input sequence?

A. LoRA (Low-Rank Adapters)

B. P-Tuning (Prompt Tuning)

C. Prefix Tuning

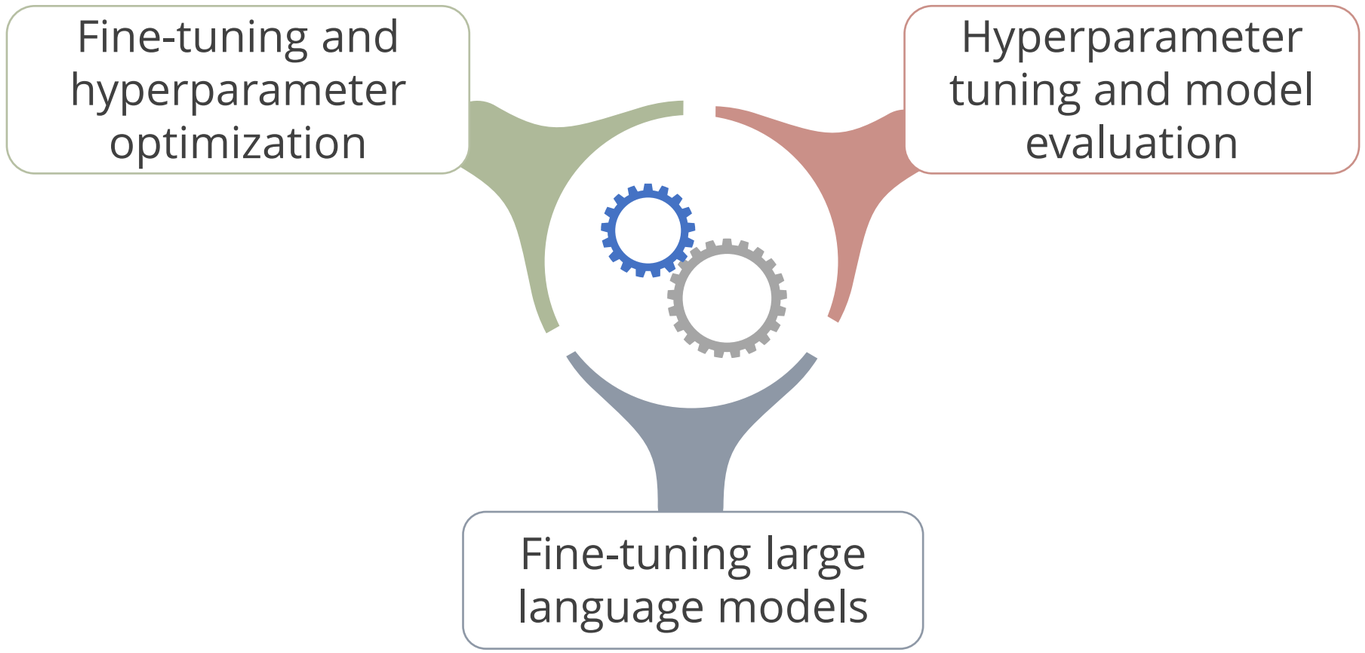
D. Adapters

Evaluation of Fine-Tuning Model

Evaluation of Fine-Tuning Model

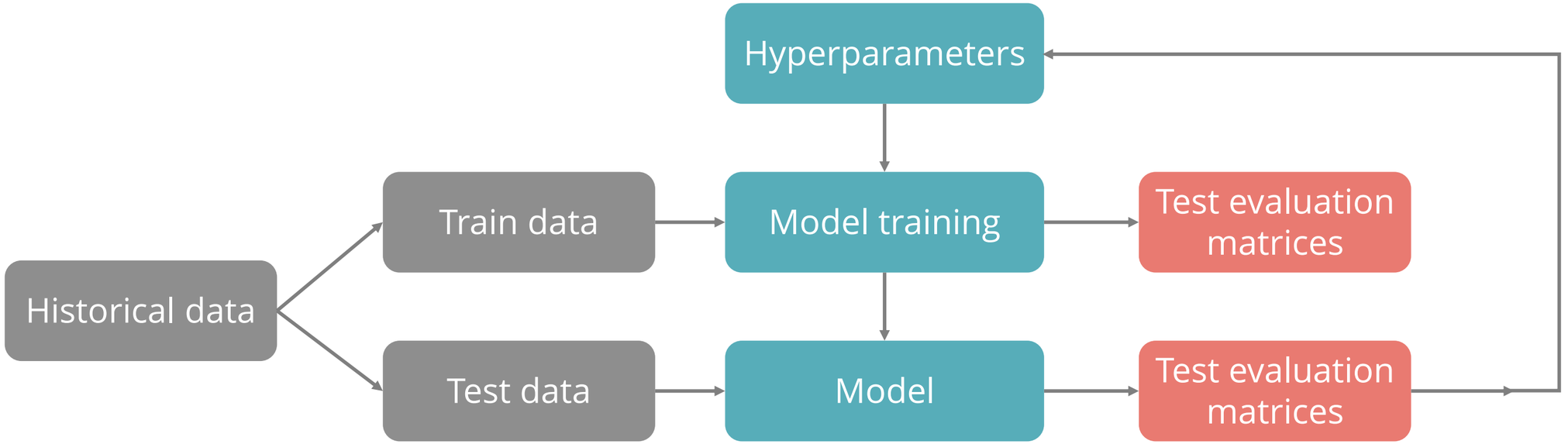
The evaluation of a fine-tuned model is a critical step in determining its performance and effectiveness for a specific task or domain.

Here are some important considerations for evaluating fine-tuned models:



Hyperparameter Tuning and Model Evaluation

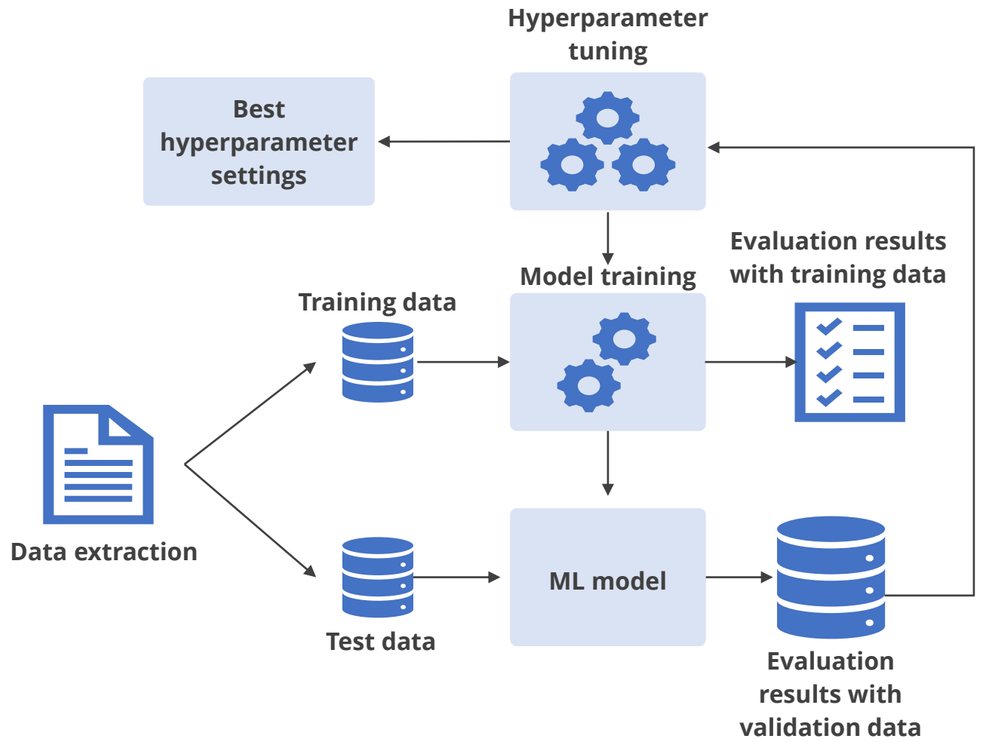
Hyperparameter tuning can have a significant impact on a model's performance, but it may only result in a marginal gain over the original metric.



The quality of the dataset is a crucial factor that often leads to better metrics and results.

Fine-Tuning and Hyperparameter Optimization

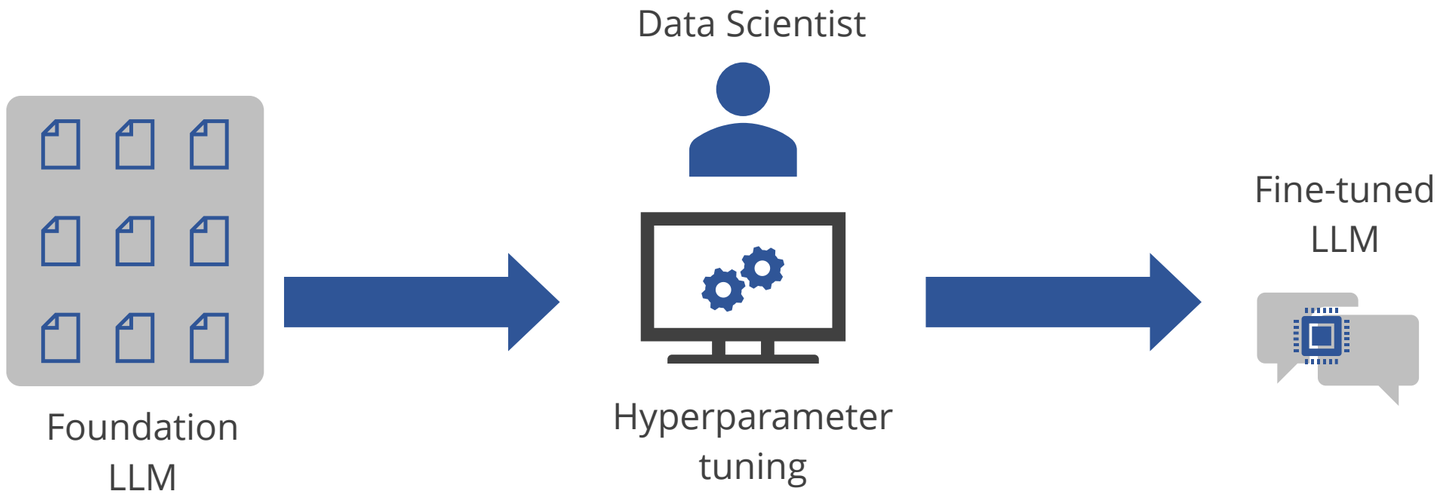
Fine-tuning models involves the meticulous process of refining a pretrained model to better align with a specific task.



Hyperparameter optimization plays a vital role in elevating the performance of a model, as even a slight tweak can make a difference in the model's outcome.

Fine-Tuning Large Language Models

Fine-tuning LLMs is the process of adjusting the parameters of a pretrained model to a specific task or domain.



Making the model learn from specific data improves accuracy. Different methods adjust the model for specific needs, making it perform better.

Quick Check

What is a crucial step in determining the performance and effectiveness of a fine-tuned model for a specific task or domain?

A. Initial training

B. Model deployment

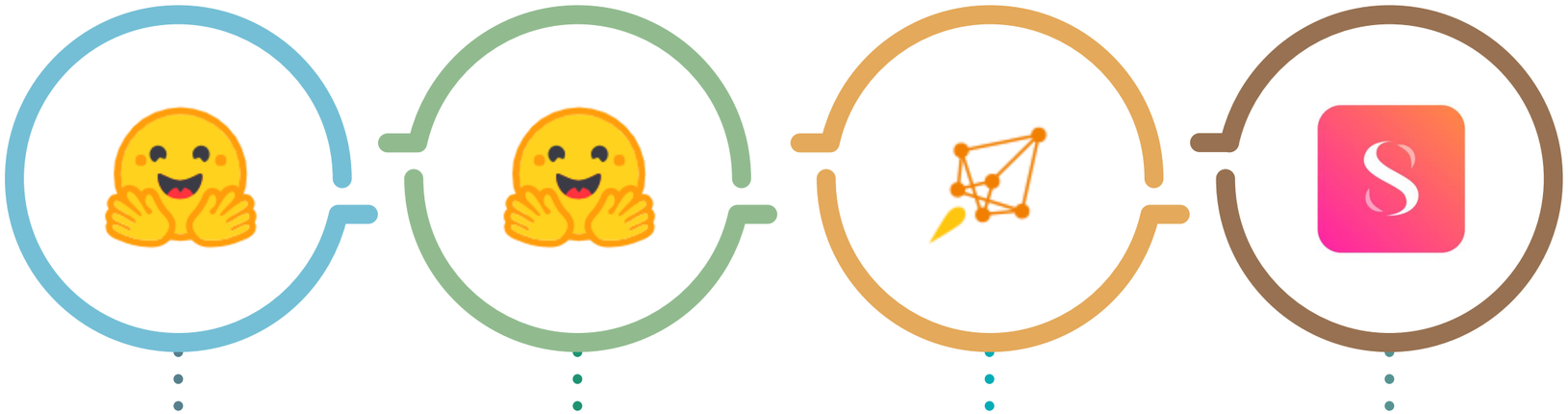
C. Hyperparameter tuning and model evaluation

D. Fine-tuning large language models

Hands-on Fine-Tuning

LLM Fine-Tuning Frameworks

The leading LLM fine-tuning frameworks and libraries that elevate language models' performance and adaptability for specific tasks and industries are:



HuggingFace Accelerate

HuggingFace Transformers

DeepSpeed

SuperAnnotate

HuggingFace Accelerate

It is another popular library for model-parallel training and inference, which can be used in conjunction with DeepSpeed for fine-tuning LLMs.

It provides an efficient and scalable solution for training and deploying large-scale models.

HuggingFace Transformers

It offers a comprehensive framework for fine-tuning LLMs, including support for LoRA and other parameter-efficient fine-tuning techniques.

The platform provides pretrained models, datasets, and tools for fine-tuning, making it easy for developers to adapt models to specific tasks.

DeepSpeed

It is a high-performance computing library that can be used for distributed training and model-parallel training, which is particularly useful for fine-tuning large models like Code Llama.

It integrates well with HuggingFace's Transformers library, allowing users to leverage its capabilities for fine-tuning.

SuperAnnotate

It is a platform that offers tools and services for fine-tuning LLMs, including data preparation, hyperparameter tuning, and model evaluation.

It helps users optimize their models for specific tasks and domains, ensuring accurate and contextually relevant outputs.

Quick Check

Which framework provides comprehensive support for fine-tuning Large Language Models (LLMs) and includes features for parameter-efficient techniques such as LoRA?

A. TensorFlow

B. PyTorch

C. Hugging face transformers

D. Keras

Fine-Tuning Best Practices

Fine-Tuning Best Practices

Here are a few tips for refining and perfecting skills with fine-tuning:

Data preparation

Label and format data consistently, employing augmentation techniques for a diverse training dataset.

Model selection

Pick tools like HuggingFace or DeepSpeed to optimize fine-tuning.

Monitor overfitting

Watch for overfitting in small datasets; use techniques like early stopping.

Fine-Tuning Best Practices

Here are a few tips for refining and perfecting skills with fine-tuning:

Hyperparameter tuning

Use Bayesian or hyperband optimization to find optimal hyperparameters efficiently.

Bias awareness

Watch for biases in pretrained models and use techniques to mitigate them.

Model evaluation

Evaluate the model using metrics to gauge effectiveness for the intended application.

Common Biases with LLM Fine-Tuning

Pretrained models may have built-in biases in what they've learned, which can continue during the fine-tuning process.

Spurious correlations and underrepresentation

Biases in data, like false connections or uneven representation, can carry over during fine-tuning.

Fairness and inductive biases

To predict a feature's impact, consider how easily it can be extracted after pretraining and the evidence during fine-tuning.

Demo: Finetune Falcon-1B

Duration: 15 minutes

In this demo, you will fine-tune the Falcon3-1B-Base using Parameter-Efficient Fine-Tuning (PEFT) with LoRA. You will tokenize a subset of WikiText-2 and configure key LoRA parameters (rank, scaling, dropout) for efficient training.

Note

Please download the solution document from the Reference Material Section and follow the Jupyter Notebook for step-by-step execution.

DEMONSTRATION

Quick Check

In the context of fine-tuning Large Language Models (LLMs), which of the following statements is correct?

A. RLHF trains LLMs directly with optimized labeled data.

B. Fine-tuning hyperparameters often boosts the original metric significantly.

C. Fine-tuning LLMs don’t need to watch for biases.

D. PEFT boosts pre-trained model performance, cutting computational needs significantly.

Guided Practice

Overview

Duration: 20 minutes

In this exercise, you'll figure out how to use LangChain, a tool that helps you make computer programs with big language models, like OpenAI's DALL-E. With LangChain, you can connect the OpenAI API to DALL-E, which is a smart computer program that turns written descriptions into pictures and artwork that looks real. Your goal is to make an app that can create images based on what users type in, like sentences, drawings, or pictures they already have.

Note

Please download the solution document from the Reference Material Section and follow the Jupyter Notebook for step-by-step execution.

GUIDED PRACTICE

Key Takeaways

- Fine-tuning LLMs is like tweaking a pretrained model to fit a specific task using suitable labeled data.
- Hyperparameter adjusting is key to fine-tuning LLMs for top-notch performance; they’re set before learning and molding the model.
- Pretrained models might carry learned biases, which can stick around during fine-tuning.
- RLHF refines LLMs by training them with human feedback, helping them follow instructions and provide quality responses.

Q&A