Post-Training Small Language Models

Stefano Fiorucci

AI/Software Engineer

💙 Language Models, Open Source and knowledge sharing

👨‍💻 I work at deepset on the Haystack LLM orchestration framework 🏗️

 anakin87

 anakin87

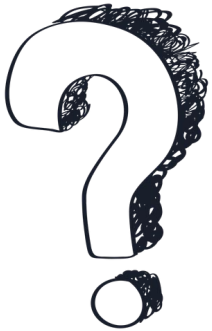
 stefano-fiorucci

Why fine-tuning Language Models?

🏢 There may be good reasons for companies

For a practitioner:

- 🔍 Get a better understanding of how LMs work
- 🎢 It's fun!



Agenda

🌱 Intro

👣 Common Post Training steps:
SFT and Preference Alignment

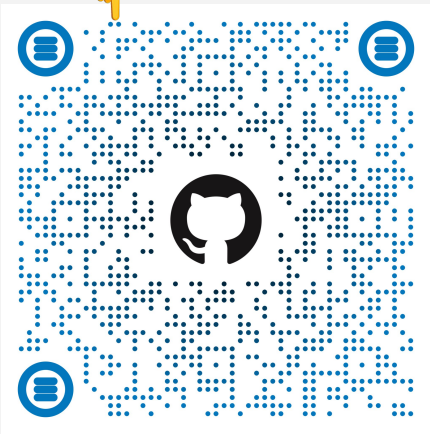
⚙️💰 Memory-efficient training

🧩 Model merging

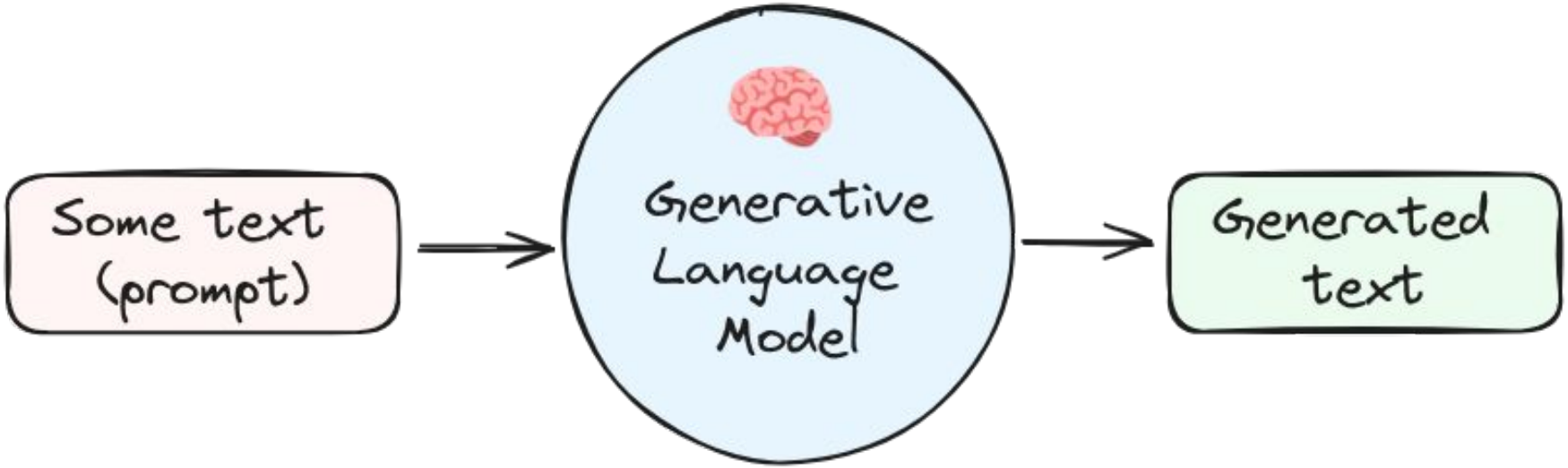
🧠💭 Reasoning models and GRPO

📱 Small Language Models on a phone

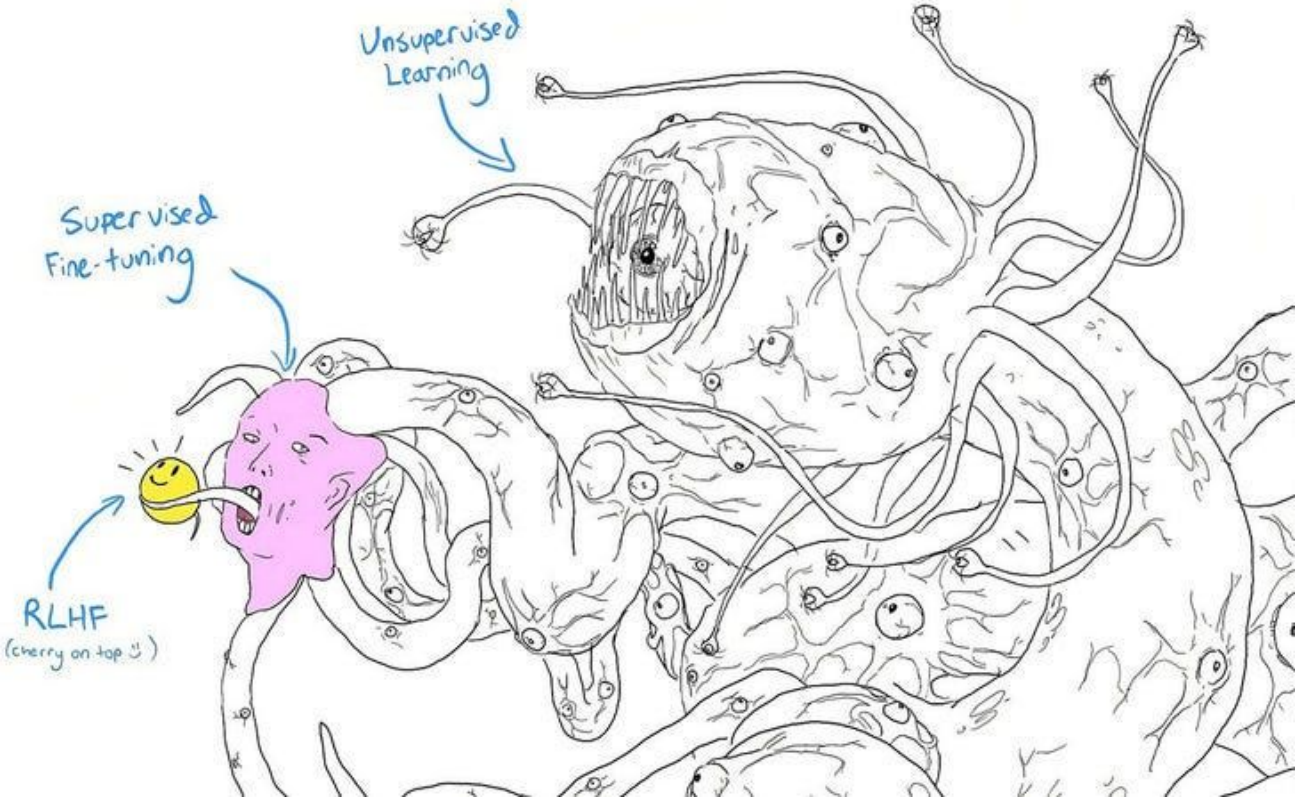
Materials 👇



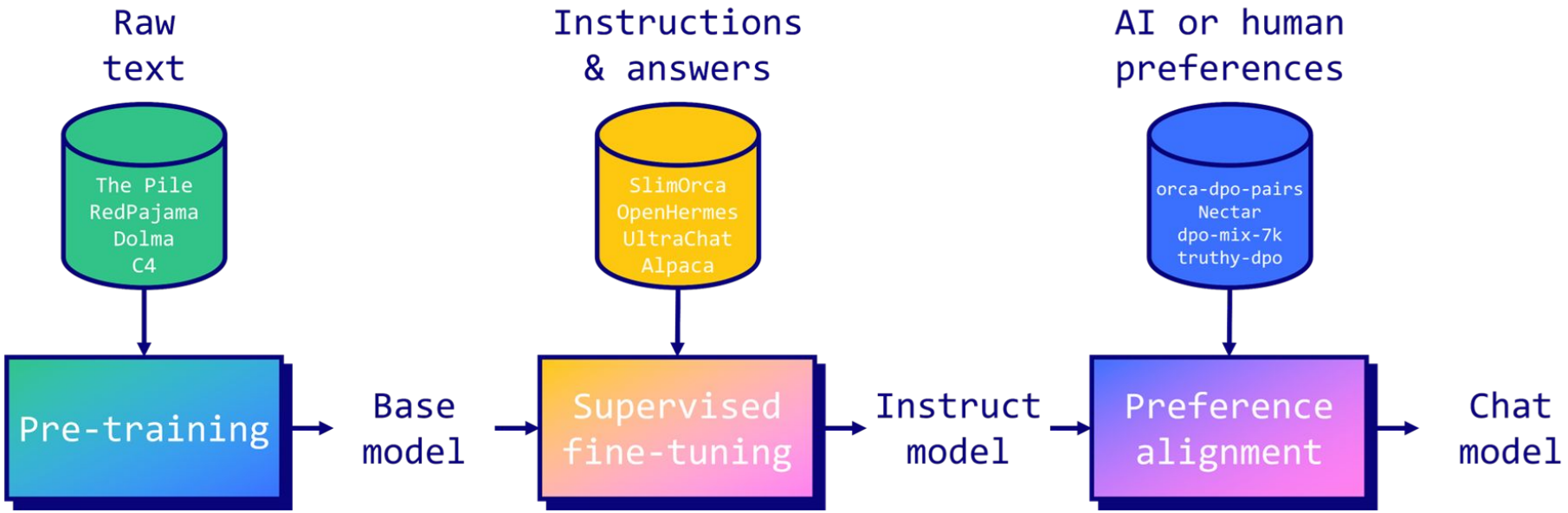
What is a Language Model?



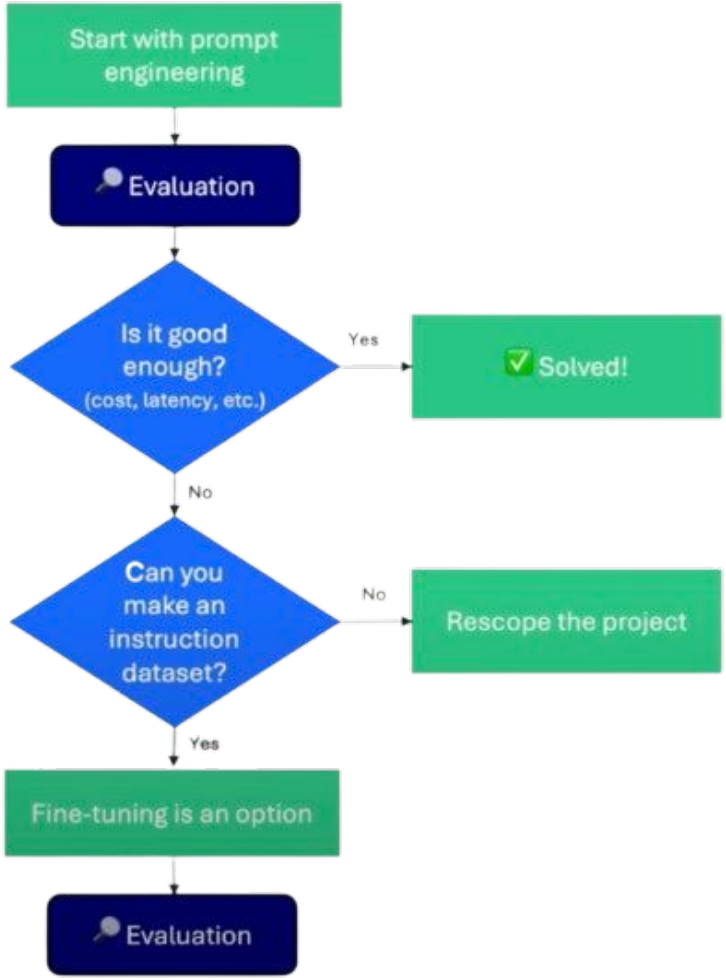
How are Language models trained?



How are Language models trained?



Source: Maxime Labonne



Do I need fine-tuning?

Source: Maxime Labonne

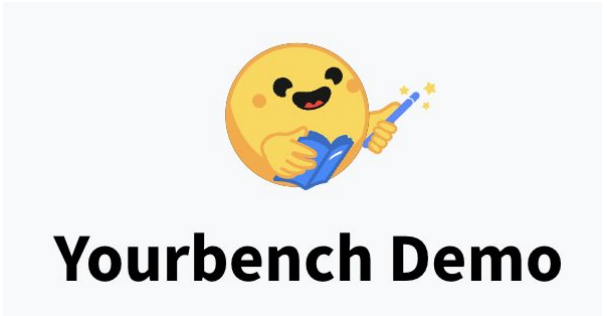
Evaluation ⚖️

“You can’t improve what you don’t measure”

- popular benchmarks

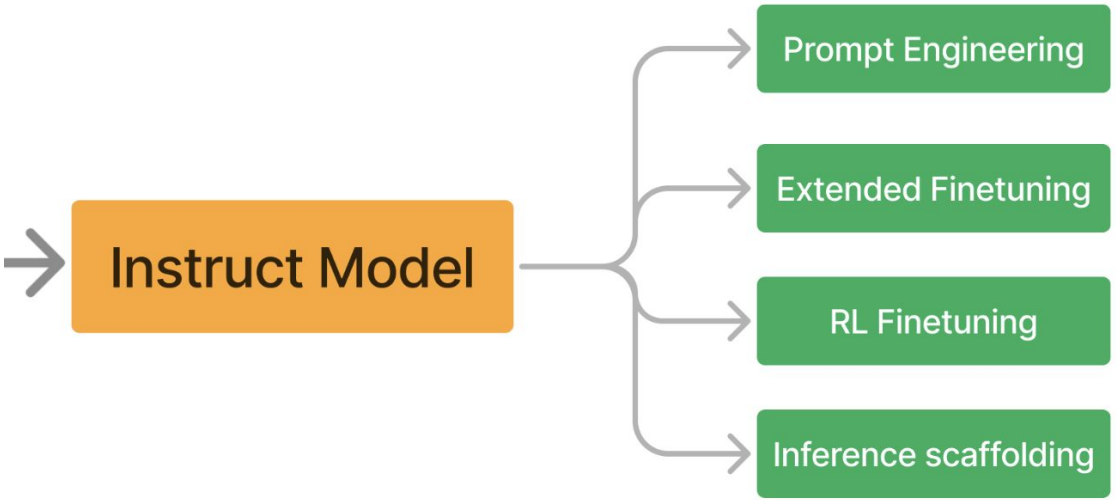
lm-evaluation-harness

- build your own benchmark



Choosing the model to train

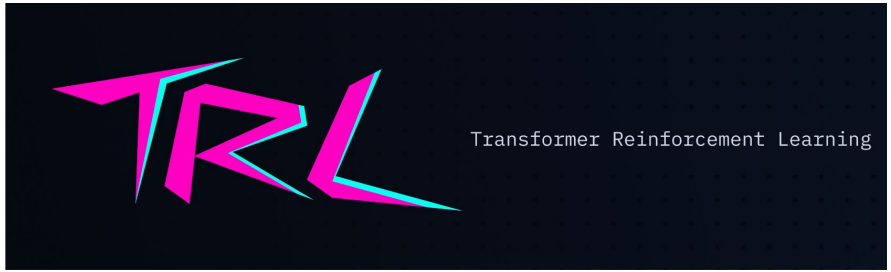
- Base vs instruct/chat



- Is this model is often chosen for fine-tuning?

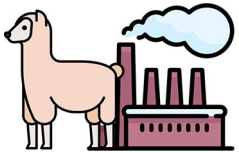
Source: Nathan Lambert

Training libraries



Axolotl

unsloth



LLaMA-Factory

Easy and Efficient LLM Fine-Tuning

volcengine/**verl**

👣 Common Post Training steps:

Supervised Fine-Tuning and Preference Alignment

Supervised Fine-Tuning (SFT) - Data

🧑 **Prompt**/Instruction/Question/User message

```
Explain the basic principles of sound waves, including their
properties and how they travel through different mediums.
```

🤖 **Response**/Completion/Answer/Assistant message

```
Sound waves are a type of mechanical wave that propagate through
a medium by causing the particles in the medium to vibrate. These
vibrations transfer energy from one particle to another, allowing
the sound wave to travel through the medium. The basic principles
of sound waves include their properties and how they travel
through different mediums…
```

Example dataset: mlabonne/FineTome-100k

```python
from datasets import load_dataset
from trl import SFTConfig, SFTTrainer

dataset = load_dataset("efederici/capybara-claude-15k-ita")

cfg = SFTConfig(
    output_dir='./mymodel',
    max_seq_length=2048,
    packing=True,
    num_train_epochs=2,
    lr_scheduler_type="cosine",
    warmup_ratio=0.2,
    learning_rate=5.0e-06,
    per_device_train_batch_size=8,
)

sft_trainer = SFTTrainer(
    model="microsoft/Phi-3.5-mini-instruct",
    args=cfg,
    train_dataset=dataset["train"],
)

sft_trainer.train()
```

Supervised Fine-Tuning (SFT) with TRL

SFT projects

mlabonne.github.io/blog/posts/Fine_Tune_Your_Own_Llama_2_Model_in_a_Colab_Notebook.html

hf.co/blog/anakin87/spectrum

Preference Alignment: PPO

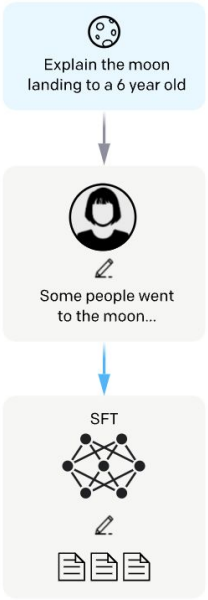
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



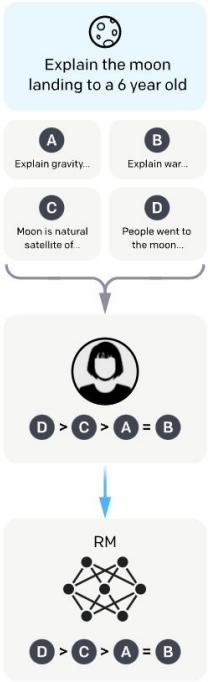
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

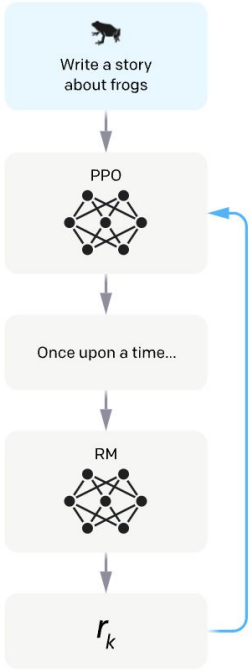
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

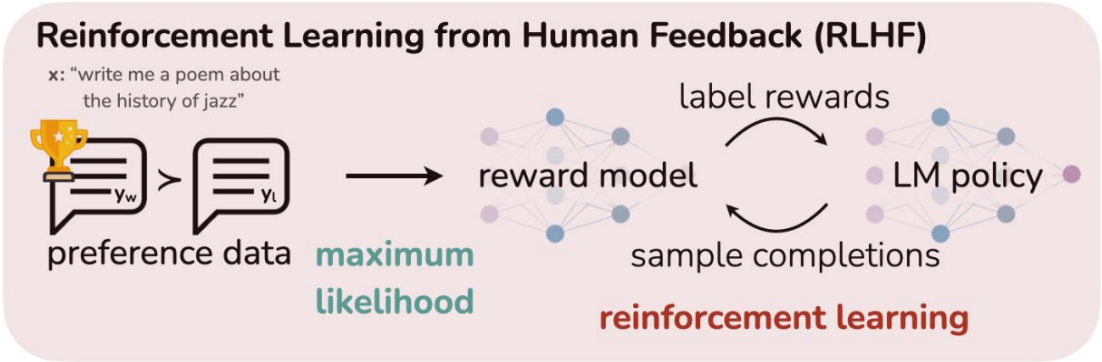
The reward model calculates a reward for the output.

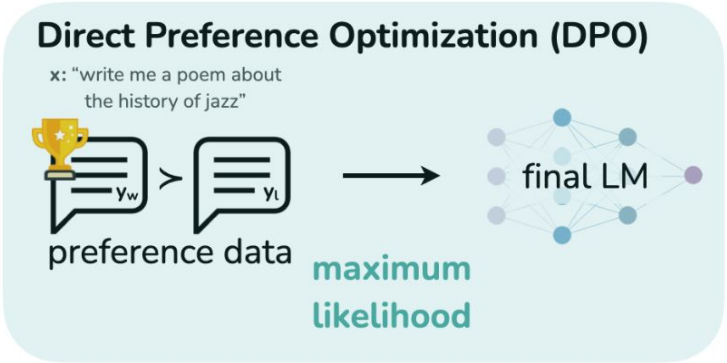
The reward is used to update the policy using PPO.



Preference Alignment: DPO

PPO





Direct Preference Optimization (DPO) - Data

🧑 Prompt

```
I would like to have some information about the X chromosome.
```

👍 Chosen Response

```
The X chromosome is one of the two sex chromosomes, with the Y
chromosome determining male sex.…
```

👎 Rejected Response

```
The X chromosome is characterized by:
1. Existence of two types: There are X chromosomes and Y chromosomes.…
```

Example dataset: mlabonne/orpo-dpo-mix-40k

```python
from datasets import load_dataset
from trl import DPOConfig, DPOTrainer

dataset = load_dataset("mlabonne/orpo-dpo-mix-40k")

cfg = DPOConfig(
    output_dir='./mymodel',
    max_length=1024,
    num_train_epochs=1,
    lr_scheduler_type="cosine",
    warmup_ratio=0.1,
    learning_rate=5.0e-06,
    per_device_train_batch_size=1,
)

dpo_trainer = DPOTrainer(
    model="google/gemma-2-2b-it",
    args=cfg,
    train_dataset=dataset["train"],
)

dpo_trainer.train()
```

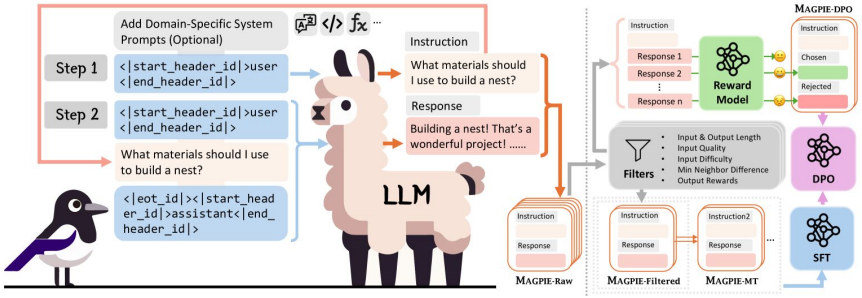
Direct Preference Optimization (DPO) with TRL

DPO projects

DPO only
mlabonne.github.io/blog/posts/Fine_tune_Mistral_7b_with_DPO.html

SFT + DPO
kaggle.com/code/anakin87/post-training-gemma-for-italian-and-beyond

Tips from practice

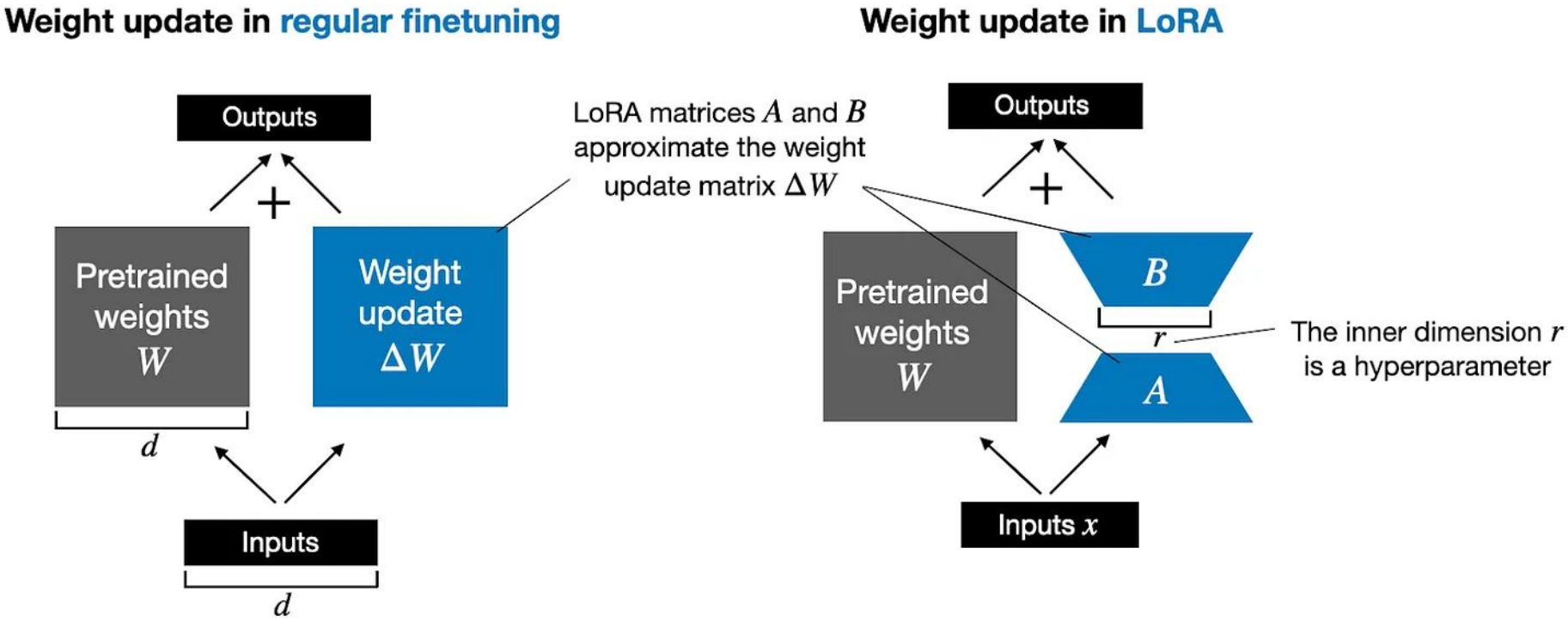


- 🗂️ Data, synthetic data
  - ⚗️ Distilabel as an inspiration
- 🧑‍🔬 Data preparation
  - chat templates
  - max_seq_length, max_prompt_length, max_length…
- 💾❌ No space left on device
- 🔥 Where to find GPU?

PRIME Intellect

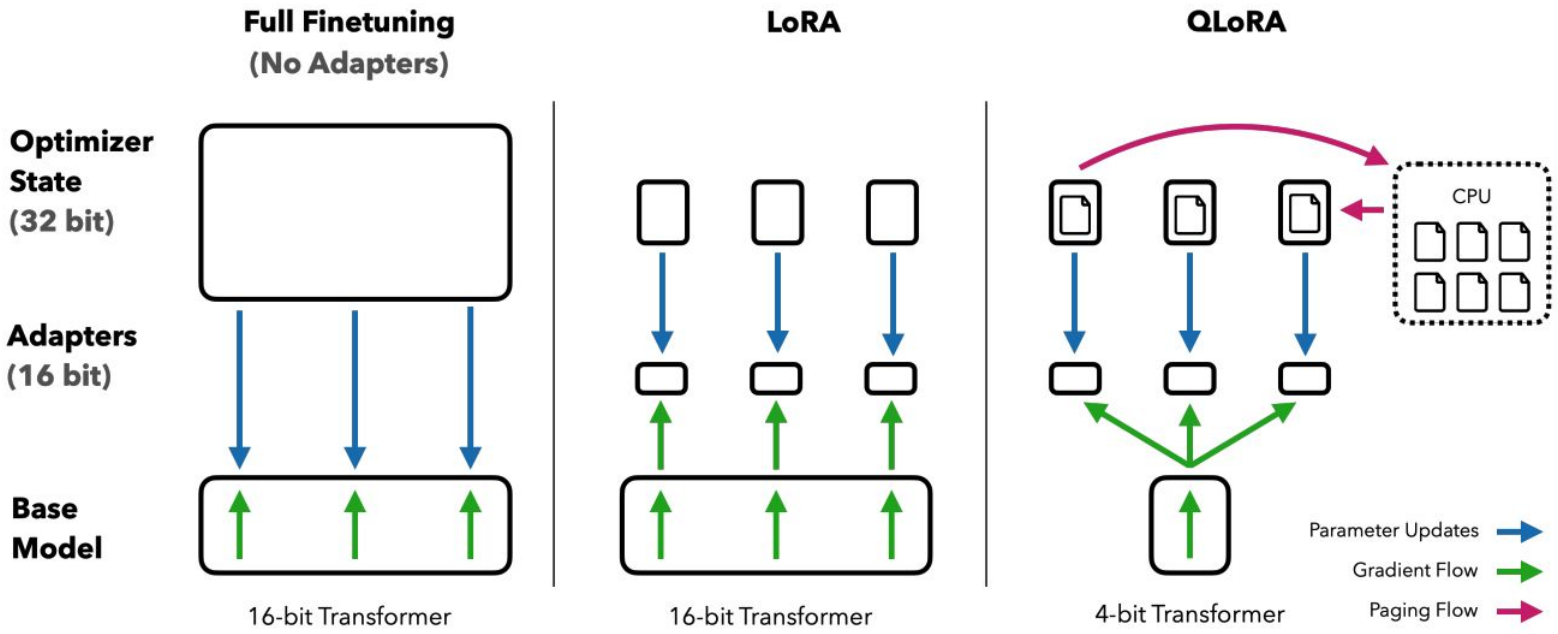
⚙️💰 Memory-efficient training

Memory-efficient training: LoRA



Source: Sebastian Raschka

Memory-efficient training: QLoRA

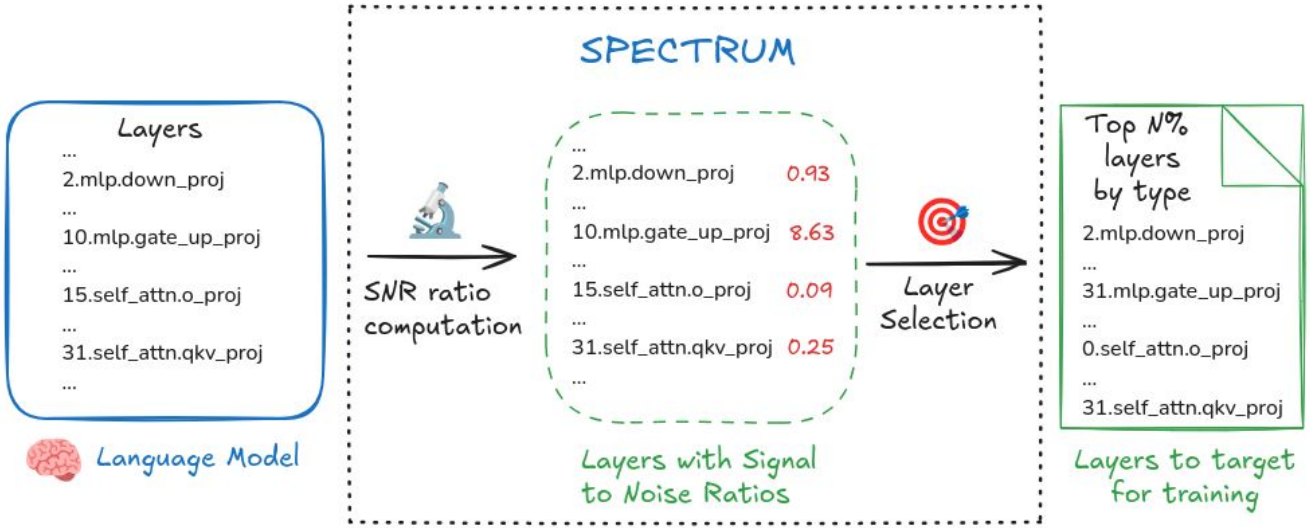


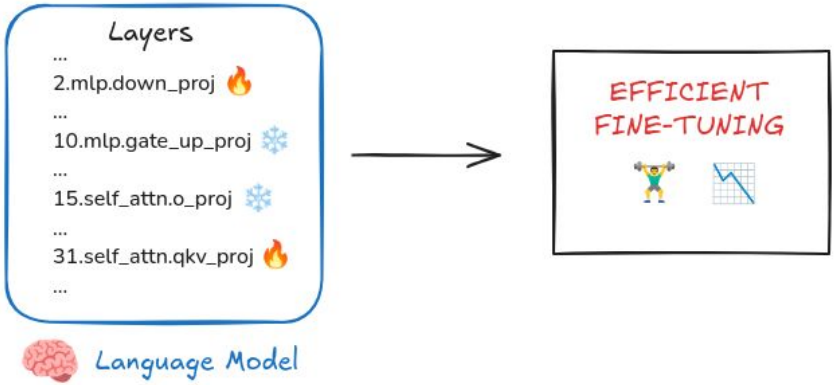
Fine-Tuning LLaMA-65B: Full - 780 GB VRAM; QLoRA - 48 GB VRAM

```python
from datasets import load_dataset
import torch
from transformers import AutoTokenizer,
AutoModelForCausalLM, BitsAndBytesConfig
from trl import SFTConfig, SFTTrainer
from peft import LoraConfig

dataset = load_dataset(...)

# quantization config
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)

model_id = "..."

model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    attn_implementation="flash_attention_2",
    torch_dtype=torch.bfloat16,
    quantization_config=bnb_config
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
```

```python
# LoRA config
peft_config = LoraConfig(
    lora_alpha=64,
    lora_dropout=0.05,
    r=64,
    bias="none",
    target_modules="all-linear",
    task_type="CAUSAL_LM",
)

cfg = SFTConfig(...)

sft_trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    peft_config=peft_config,
    args=cfg,
    train_dataset=dataset["train"],
)

sft_trainer.train()
```

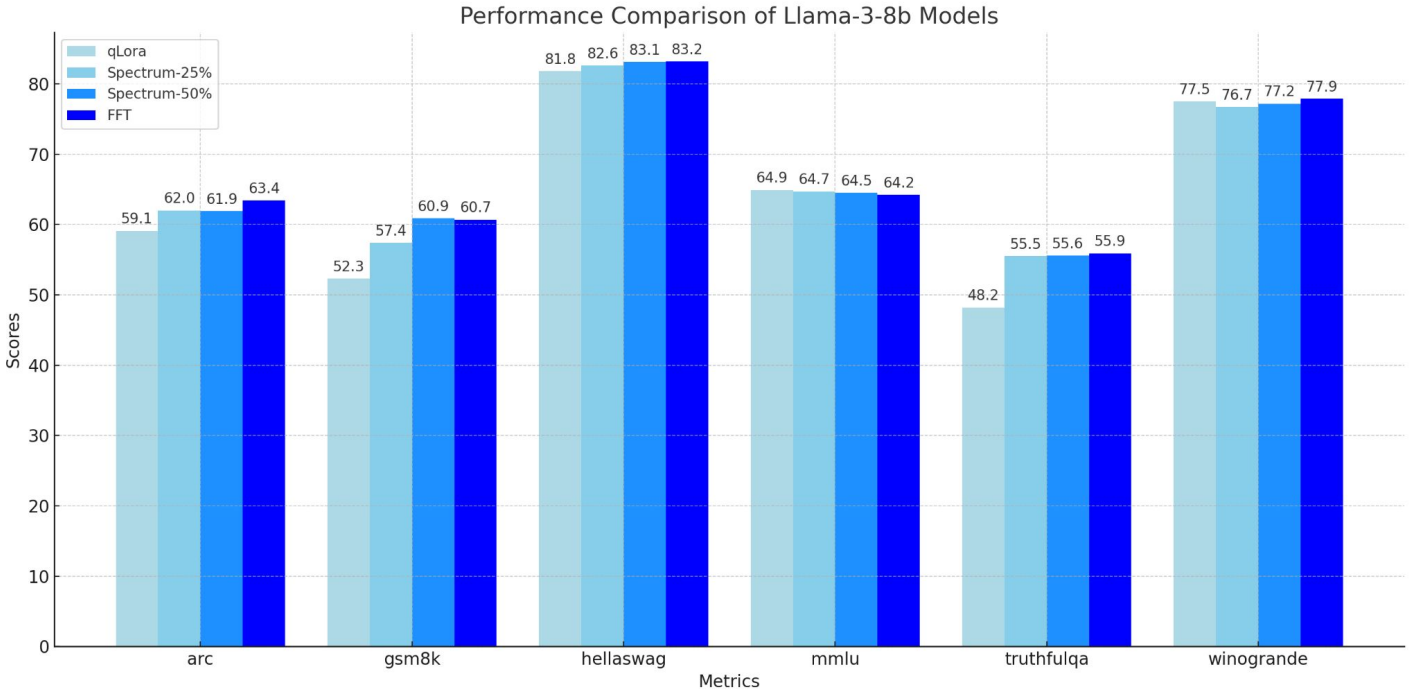
QLoRA with TRL





Memory-efficient training: Spectrum

QLoRA vs Spectrum



Memory efficiency

QLoRA is better on a single GPU; Spectrum in distributed training

```python
from datasets import load_dataset
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from trl import SFTConfig, SFTTrainer
from peft import LoraConfig

# 1. USE SPECTRUM TO IDENTIFY PARAMETERS TO TRAIN
# python spectrum.py --model-name <model-name> --top-percent
<top % of snr ratios to target>

with open("snr_results.yaml", "r") as fin:
    yaml_parameters = fin.read()

unfrozen_parameters = []
for line in yaml_parameters.splitlines():
  if line.startswith("- "):
    unfrozen_parameters.append(line.split("- ")[1])

dataset = load_dataset(...)

model_id = "..."
```

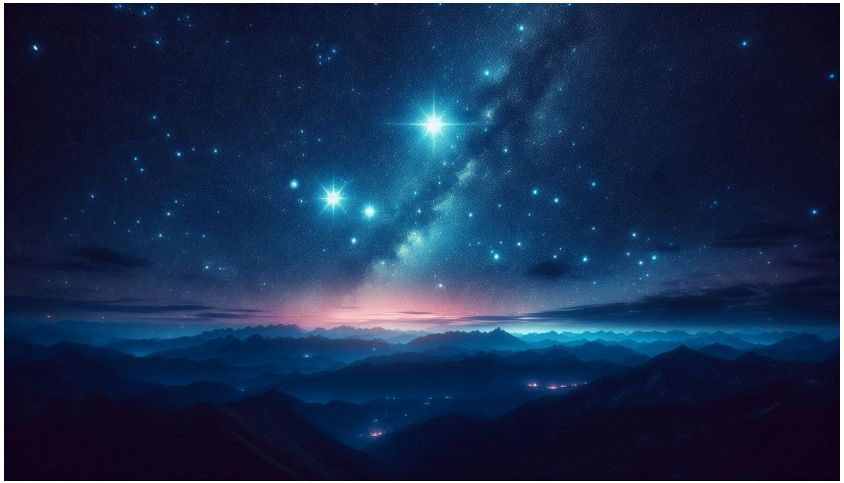
```python
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    attn_implementation="flash_attention_2",
    torch_dtype=torch.bfloat16,
)
tokenizer = AutoTokenizer.from_pretrained(model_id)

# 2. FREEZE ALL PARAMETERS EXCEPT THOSE IDENTIFIED BY SPECTRUM
freeze_and_unfreeze_parameters(model, unfrozen_parameters)

cfg = SFTConfig(...)

sft_trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    args=cfg,
    train_dataset=dataset["train"],
)
sft_trainer.train()
```

Spectrum with TRL

Projects on memory-efficient training

QLoRA

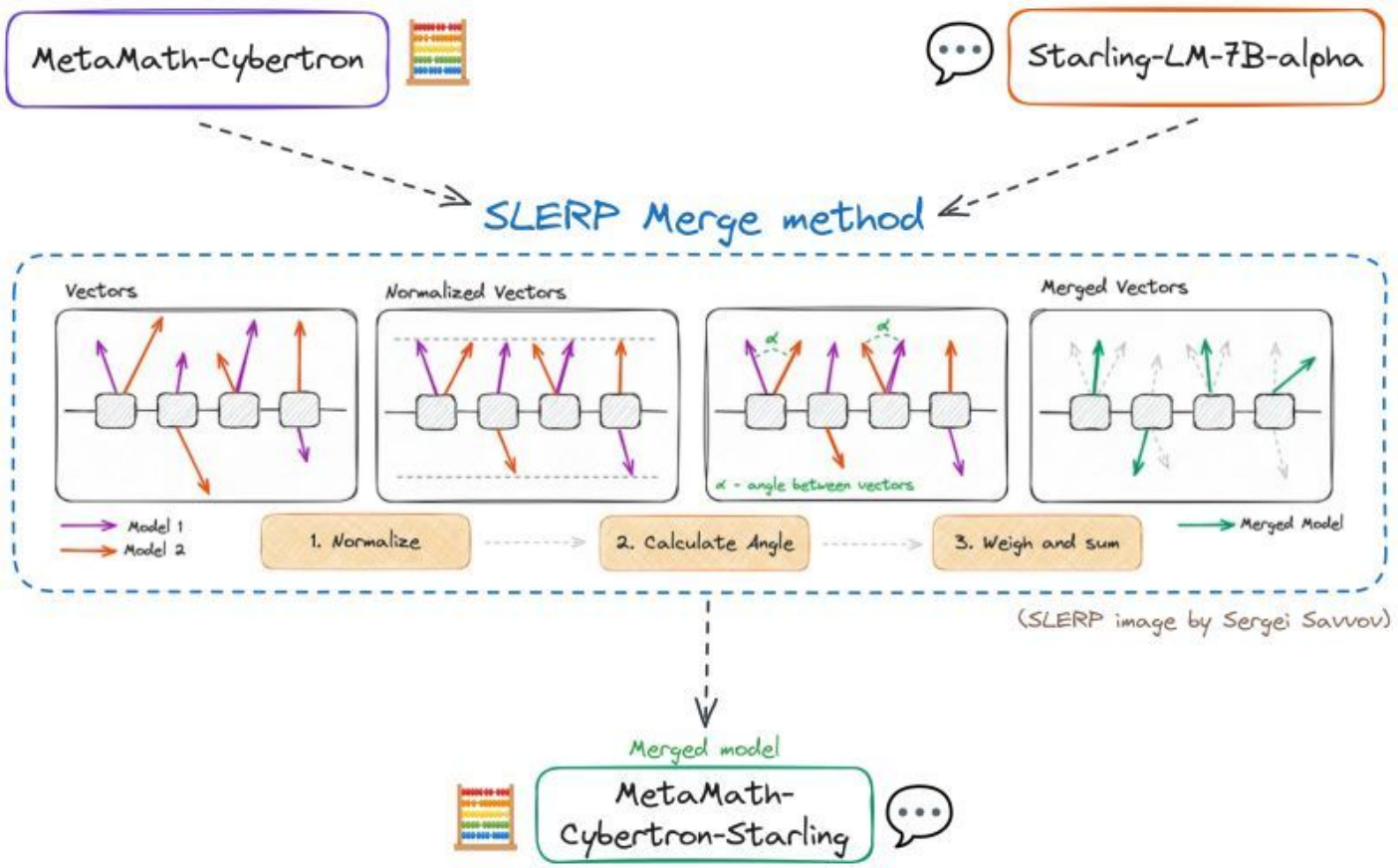
🦙
https://www.philschmid.de/fine-tune-llms-in-2024-with-trl



💎
https://ai.google.dev/gemma/docs/core/huggingface_text_finetune_qlora

Spectrum
hf.co/blog/anakin87/spectrum

Model merging



```
yaml_config = """
slices:
  - sources:
      - model: AIDC-ai-business/Marcoroni-7B-v3
        layer_range: [0, 32]
      - model: EmbeddedLLM/Mistral-7B-Merge-14-v0.1
        layer_range: [0, 32]
merge_method: slerp
base_model: AIDC-ai-business/Marcoroni-7B-v3
parameters:
  t:
    - filter: self_attn
      value: [0, 0.5, 0.3, 0.7, 1]
    - filter: mlp
      value: [1, 0.5, 0.7, 0.3, 0]
    - value: 0.5
dtype: bfloat16
"""

with open('config.yaml', 'w', encoding="utf-8") as f:
    f.write(yaml_config)

! mergekit-yaml config.yaml merge --copy-tokenizer --allow-
crimes --out-shard-size 1B --lazy-unpickle
```

Model merging with Mergekit

arcee-ai/**mergekit**

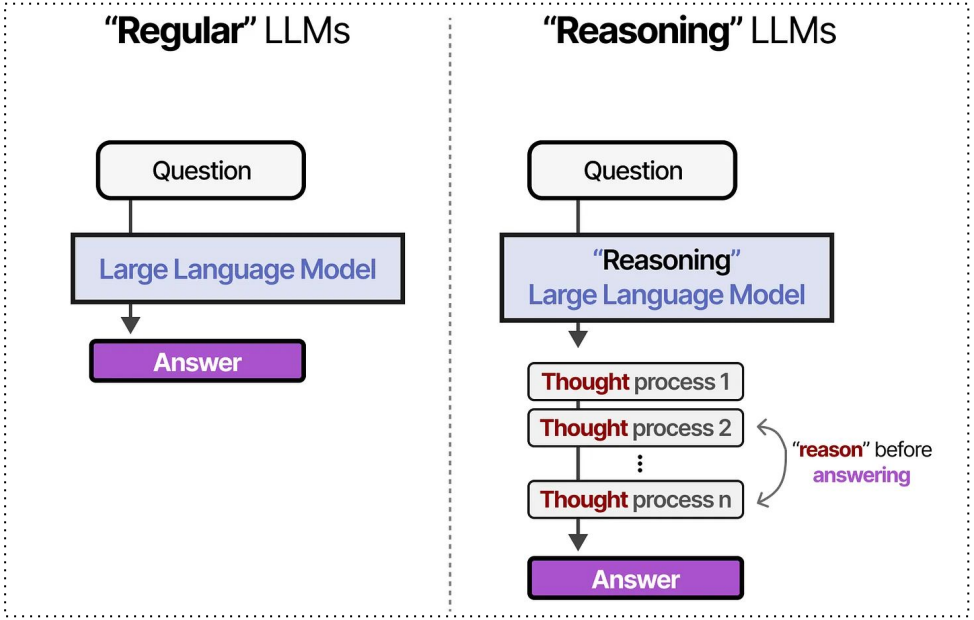
Tools for merging pretrained large language models.

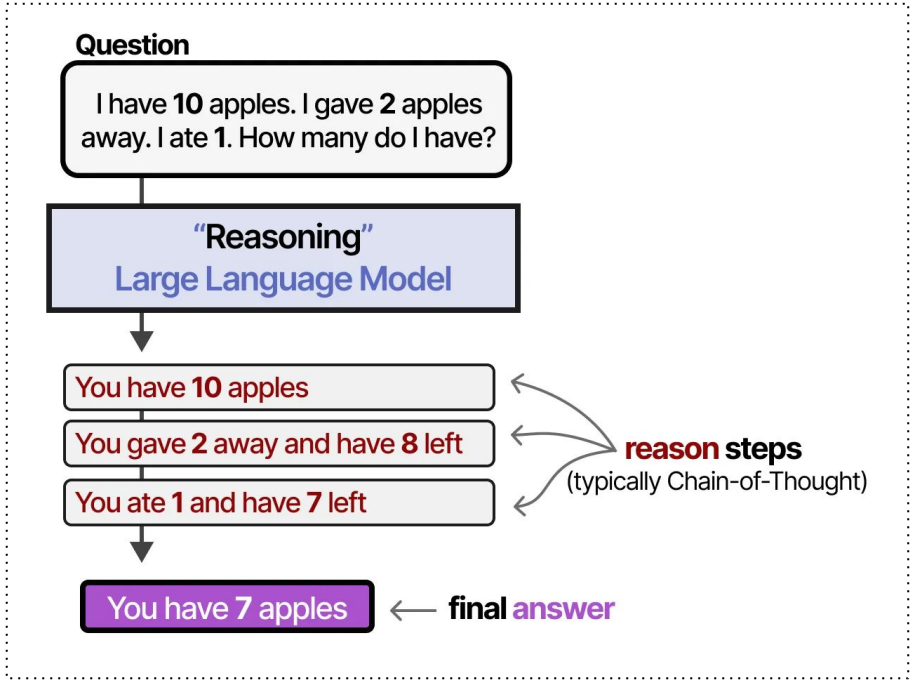
Model merging project

https://mlabonne.github.io/blog/posts/2024-01-08_Merge_LLMs_with_mergekit.html

Reasoning models and GRPO

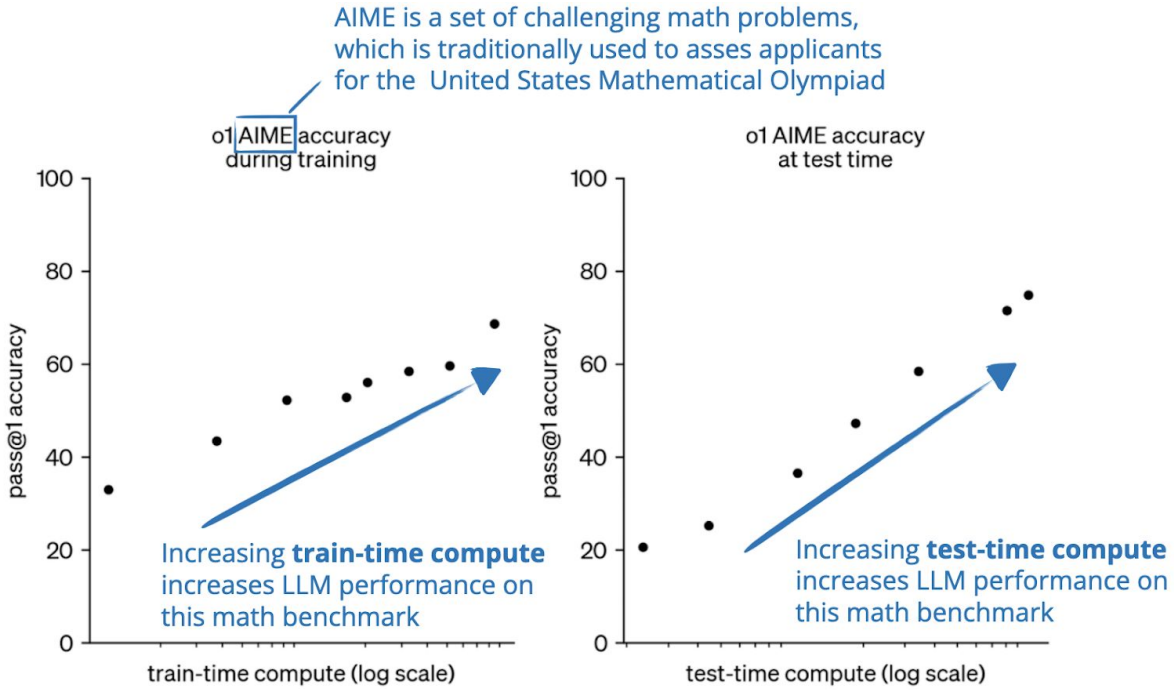
Reasoning Models





Source: Maarten Grootendorst

Reasoning Models: why and how



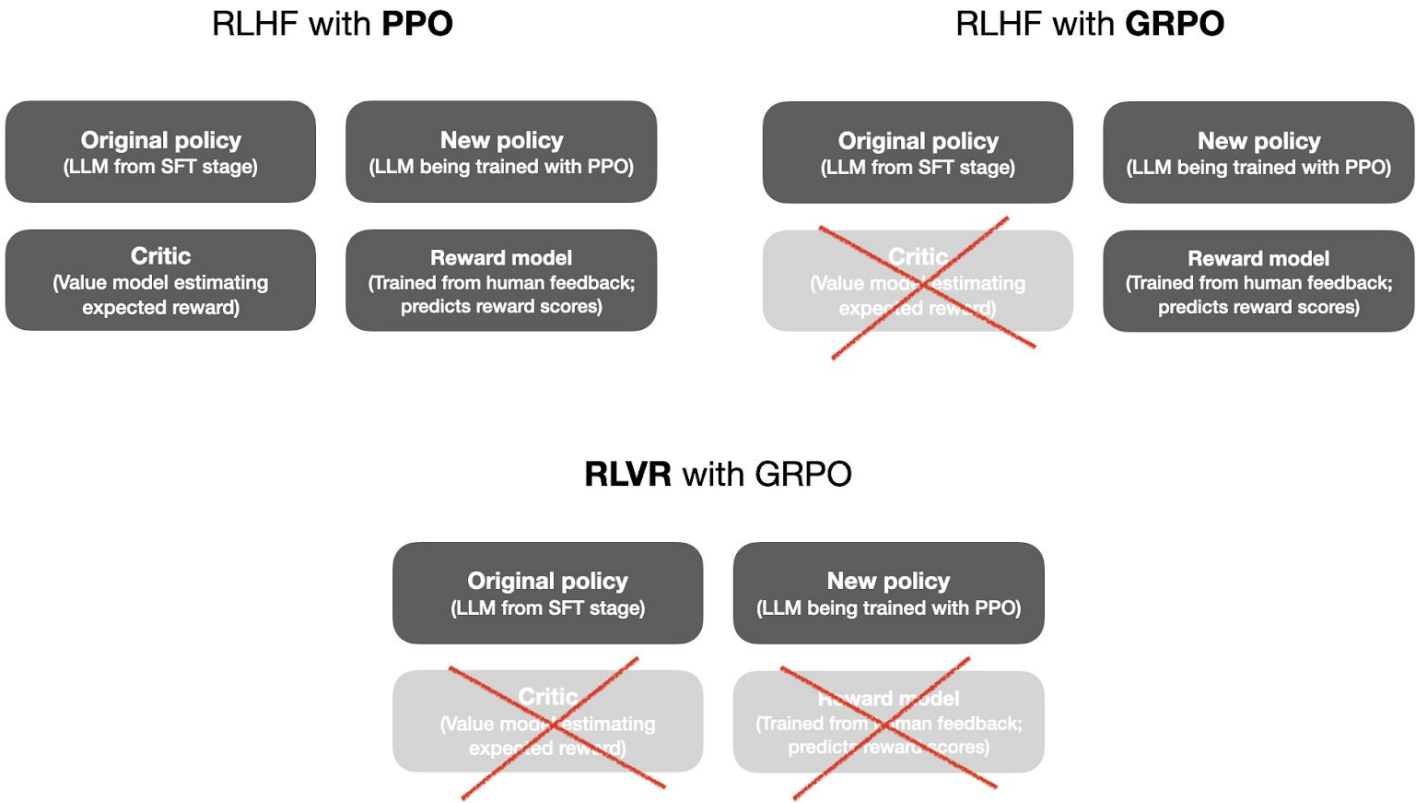
o1 performance smoothly improves with both train-time and test-time compute

Source: Sebastian Raschka

RLVR with GRPO 🐋

1. The model generates a **group** of responses via sampling.
2. Each response is evaluated using **deterministic reward functions**.
3. An average score is calculated across the group.
4. Individual response scores are compared against this average.
5. The model is updated to favor higher-scoring responses.

GRPO vs PPO



Source: Sebastian Raschka

GRPO - Data

Question ❓
Visible to the model

```
Weng earns $12 an hour for babysitting. Yesterday, she just did 50
minutes of babysitting. How much did she earn?
```

Answer ❗
***Not** visible to the model; used only for reward computation*

`10`

Example dataset: openai/gsm8k

GRPO - Reward functions

```python
def correctness_reward_func(prompts, completions, answer, **kwargs) -> list[float]:
    """Reward function that checks if the completion is correct."""

    responses = [completion[0]['content'] for completion in completions]
    q = prompts[0][-1]['content']
    extracted_responses = [extract_xml_answer(r) for r in responses]

    return [2.0 if r == a else 0.0 for r, a in zip(extracted_responses, answer)]


def strict_format_reward_func(completions, **kwargs) -> list[float]:
    """Reward function that checks if the completion has a specific format."""

    pattern = r"^<reasoning>\n.*?\n</reasoning>\n<answer>\n.*?\n</answer>\n$"
    responses = [completion[0]["content"] for completion in completions]
    matches = [re.match(pattern, r, flags=re.DOTALL) for r in responses]

    return [0.5 if match else 0.0 for match in matches]
```

Source: William Brown

```python
from datasets import load_dataset
from trl import GRPOConfig, GRPOTrainer

dataset = load_dataset(...)

# Define the reward functions
...

training_args = GRPOConfig(num_generations=16, ...)

trainer = GRPOTrainer(
    model="Qwen/Qwen2-0.5B-Instruct",
    reward_funcs=[correctness_reward_func,
                  strict_format_reward_func],
    args=training_args,
    train_dataset=dataset,
)

trainer.train()
```

GRPO with TRL

GRPO projects

willccbb / grpo_demo.py
Last active yesterday
GRPO Llama-1B

1 file 379 forks 110 comments 1224 stars

GRPO Llama-1B (GSM8K)

https://gist.github.com/willccbb/4676755236bb08cab5f4e54a0475d6fb

👑 🗓️ Qwen Scheduler GRPO

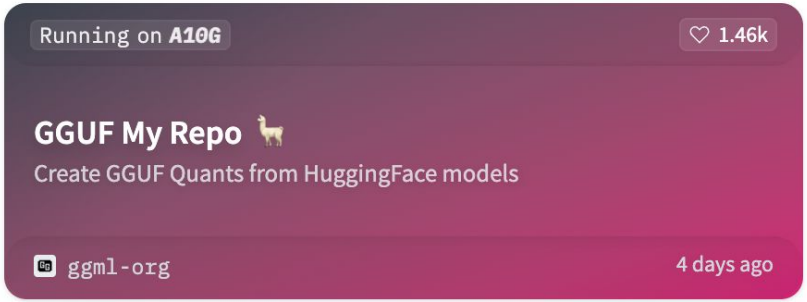
hf.co/blog/anakin87/qwen-scheduler-grpo
github.com/anakin87/qwen-scheduler-grpo

GRPO: what I've learned

- GRPO works well for verifiable task 🔢 ✅
- 🕹️ Elicitation
- Base model matters
- 🎯 Reward functions design is crucial
- "Aha moment" might be overhyped
- 🦥 Unsloth: great for saving GPU but beware

📱 Small Language Models on a phone

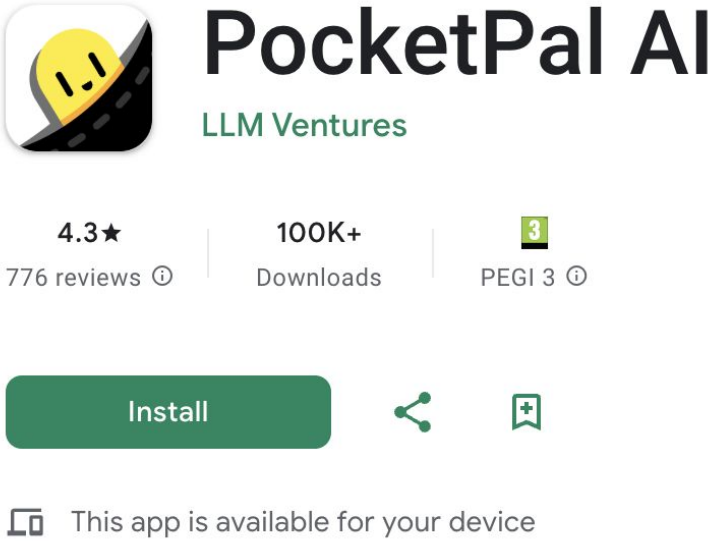
Small Language Models on a phone: GGUF



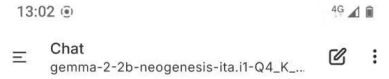
hf.co/spaces/ggml-org/gguf-my-repo

github.com/ggml-org/llama.cpp/tree/master/tools/quantize

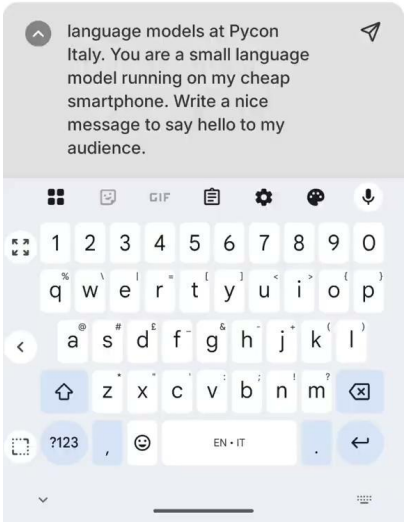
Small Language Models on a phone: PocketPal



📱 Small Language Models on a phone



Me + Nokia X10 (2022)



Post-training Small Language Models

Stefano Fiorucci

Talk Materials

GitHub

LinkedIn

anakin87

stefano-fiorucci