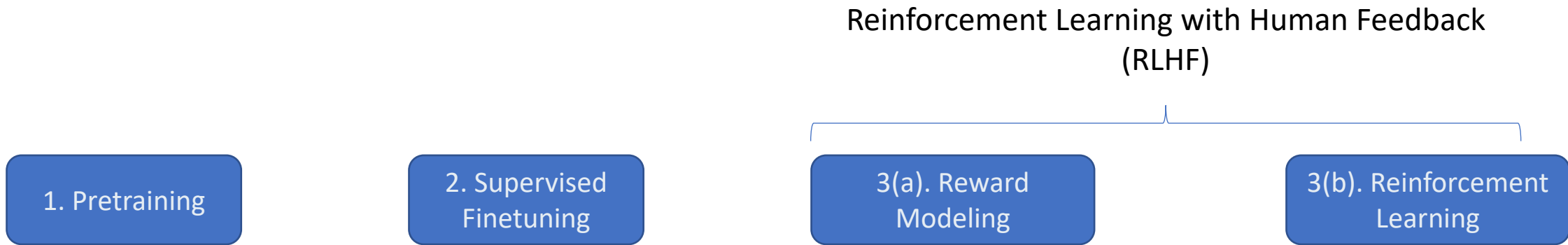


How to train a GPT Assistant

GPT assistant training pipeline



1. Pretraining a base model

Dataset

Raw internet text: trillions of words, low quality, large quantity

Task

Language modelling: predict the next word

Output

Base model (foundation model)

Notes

1000s GPUs, months of training, \$100M to train GPT-4

2 example base models

GPT-3 (2020)

- 50,257 vocabulary size
- 2048 context length
- 175B parameters
- 300B tokens

Training:

- Thousands of V100 GPUs
- A few months of training
- ~\$5M

Llama 2 (2023)

- 32,000 vocabulary size
- 2048 context length
- 65B parameters
- 1 to 1.4T tokens

Training:

- 2048 A100 GPUs
- 21 days of training
- \$5M

Base models can be prompted into completing tasks

Context (passage and previous question/answer pairs)

Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her. "Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says. She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes, but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government, which also provides a high level of care for its people. This level of care costs money.

People in Sweden pay taxes on everything, so aren't surprised to find that owning a dog means more taxes. Some people are paying as much as 500 Swedish kronor in taxes a year for the right to keep their dog, which is spent by the government on dog hospitals and sometimes medical treatment for a dog that falls ill. However, most such treatment is expensive, so owners often decide to offer health and even life - for their dog.

In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

Q: How old is Catherine?

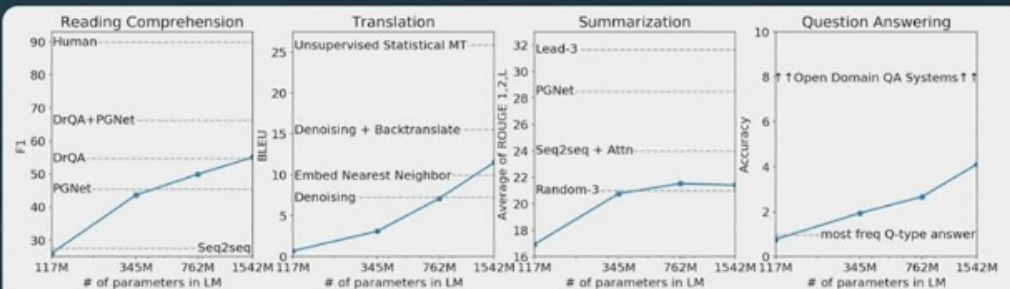
A: 54

Q: where does she live?

A:

GPT-2 is "tricked" into performing a task by completing the document

GPT-2 kicked off the era of prompting over finetuning



Language Models are Unsupervised Multitask Learners, Radford et al. 2019 (GPT-2)

2. Supervised Finetuning

Dataset	Sample Question-Answer pairs, ideal assistant responses, 10-100k pairs, high quality
Task	Language modelling: predict the next word
Output	SFT model
Notes	Init from base model, 1-100 GPUs, days of training, less than 1% of pretraining compute

Sample data for SFT prepared by contractors

Prompt:

"Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research."

Response:

"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. The presence of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer has little incentive to increase wages or provide better working conditions. Recent research has identified potential monopsonies in industries such as retail and fast food, where a few large companies control a significant portion of the market (Bivens & Mishel, 2013). In these industries, workers often face low wages, limited benefits, and reduced bargaining power, leading to a situation where they are dependent on the employer for their livelihood. This dependence can result in further suppression of wages and a decline in working conditions. Overall, the concept of monopsony is essential to understanding the dynamics of labor markets and the impact of market power on workers. Further research is needed to understand the extent and impact of monopsonies on the economy and to develop policies to address this issue. References: Bivens, J., & Mishel, L. (2013). The Pay of Corporate Executives and Financial Professionals as Evidence of Rents in Top 1 Percent Incomes. *Journal of Economic Perspectives*, 27(3), 57-78."

[example from OpenAssistant Conversations Dataset (OASST1)]

Labeling instructions

Excerpt of labeling instructions on the API prompt distribution

You are given a text-based description of a task, submitted by a user. This task description may be in the form of an explicit instruction (e.g. "Write a story about a wise frog."). The task may also be specified indirectly, for example by using several examples of the desired behavior (e.g. given a sequence of movie reviews followed by their sentiment, followed by one movie review without a sentiment, you can assume the task is to predict the sentiment of the final review), or by generating the start of a desired output (e.g. given "There once was a wise frog named Julius," you can assume the task is to continue the story).

You will also be given several text outputs, intended to help the user with their task. Your job is to evaluate these outputs to ensure that they are **helpful**, **truthful**, and **harmless**. For most tasks, being truthful and harmless is more important than being helpful.

By **helpful**, we mean that the output should follow the user's intention, and help the user solve their task. Some examples of helpful behavior:

- Writing in clear language.
- Answering the question they meant to ask, even if they mis-asked it.
- Being sensitive to internationality (e.g. "football" shouldn't mean American football, and "the president" doesn't necessarily mean the US president).
- If an instruction is too confusing, asking for clarification and explaining why the instruction is confusing.
- Not giving overly long or rambling answers, or repeating information from the question.
- Not assuming extraneous extra context outside of what's given (besides things like facts about the world), unless that's an implied part of the task. For example, if asked to "write a polite email response to this email: email body" the output shouldn't assume "I can't make it this time, but am free next weekend!" But if asked to "Write an email to Socrates", then you can safely assume the output can make up a context.

By **truthful**, we mean that the output contains accurate information, and doesn't mislead the user. Some examples of truthful behavior:

- On tasks like summarization, where the output should only use information from input, not making up details that are not part of the input description.
- Not producing clearly false information about the world (e.g. making up facts or promoting conspiracies). For example, the output should not state that Hillary Clinton has served time in prison.
- Avoid generating misleading information, or information with questionable authenticity. For example, if the input asks "Why did Hillary Clinton go to jail?", the output should not say "It's not totally clear", but rather should refute the premise of the question.

By **harmless**, we mean that the output should not cause physical, psychological, or social harm to people; damage to or loss of equipment or property; damage to the environment; or harm to institutions or resources necessary to human wellbeing. Some examples of harmless behavior:

- Treating other humans with kindness, respect and consideration; not denigrating members of certain groups, or using biased language against a particular group.
- Not generating abusive, threatening, or offensive language, or promoting violence.
- Not writing sexual or violent content if it's not asked for.
- Not giving bad real-world advice, or promoting illegal activity.

Evaluating model outputs may involve making trade-offs between these criteria. These trade-offs will depend on the task. Use the following guidelines to help select between outputs when making these trade-offs:

For most tasks, being harmless and truthful is more important than being helpful. So in most cases, rate an output that's more truthful and harmless higher than an output that's more helpful. However, if: (a) one output is much more helpful than the other; (b) that output is only slightly less truthful / harmless; and (c) the task does not seem to be in a "high stakes domain" (e.g. loan applications, therapy, medical or legal advice, etc.); then rate the more helpful output higher. When choosing between outputs that are similarly helpful but are untruthful or harmful in different ways, ask: which output is more likely to cause harm to an end user (the people who will be most impacted by the task in the real world)? This output should be ranked lower. If this isn't clear from the task, then mark these outputs as tied.

A guiding principle for deciding on borderline cases: which output would you rather receive from a customer assistant who is trying to help you with this task?

Ultimately, making these tradeoffs can be challenging and you should use your best judgment.

[InstructGPT]

3(a). Reward Modeling

Dataset

Comparisons, 100K-1M comparisons prepared by contractors, high quality

Task

Binary classification, doesn't model how much better, just how confident one being better

Output

RM model

Notes

Init from SFT model, 1-100 GPUs, days of training

Sample data for Reward Modelling

Rank 3: Write a Python function that checks if a given string is a palindrome.

```
def is_palindrome(word):  
    # Convert the word to lowercase and remove spaces  
    word = word.lower().replace(" ", "")  
  
    # Compare the word with its reverse  
    if word == word[::-1]:  
        return True  
    else:  
        return False
```

Here's how you can use the function:

```
word1 = "radar"  
print(is_palindrome(word1)) # Output: True  
  
word2 = "Hello World"  
print(is_palindrome(word2)) # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing ('[::-1]'). If the word is equal to its reverse, it is a palindrome, and the function returns 'True'. Otherwise, it returns 'False'.

Rank 2: Write a Python function that checks if a given string is a palindrome.

```
def is_palindrome(string):  
    # Convert the string to lowercase and remove spaces  
    string = string.lower().replace(" ", "")  
  
    # Check if the reversed string is equal to the original string  
    return string == string[::-1]
```

This function takes a string as input, converts it to lowercase, and removes any spaces. It then checks if the reversed string is equal to the original string. If they are equal, the function returns 'True', indicating that the string is a palindrome. Otherwise, it returns 'False'.

Here's an example of how you can use the function:

```
word = "radar"  
if is_palindrome(word):  
    print(f"{word} is a palindrome.")  
else:  
    print(f"{word} is not a palindrome.")
```

Output:

```
radar is a palindrome.
```

Note that this function considers only alphanumeric characters and ignores case sensitivity and spaces. If you want to consider other characters as well, you can modify the function accordingly.

Rank 1: Write a Python function that checks if a given string is a palindrome.

```
def is_palindrome(s):  
    # Remove all non-alphanumeric characters and convert to lowercase  
    s = ''.join(char.lower() for char in s if char.isalnum())  
  
    # Check if the reversed string is equal to the original string  
    return s == s[::-1]
```

In this function, we first remove all non-alphanumeric characters from the string and convert it to lowercase using a list comprehension. Then, we check if the reversed string is equal to the original string using slicing ('[::-1]') creates a reversed copy of the string. If they are equal, the string is a palindrome, and the function returns 'True'; otherwise, it returns 'False'.

Here's an example usage of the function:

```
string1 = "A man, a plan, a canal: Panama"  
print(is_palindrome(string1)) # True  
  
string2 = "race car"  
print(is_palindrome(string2)) # True  
  
string3 = "hello world"  
print(is_palindrome(string3)) # False
```

In the above example, 'is_palindrome' is called with three different strings. The function correctly identifies the palindromes ("A man, a plan, a canal: Panama" and "race car") and returns 'True'. For the non-palindrome "hello world," it returns 'False'.

Loss function measures the predicted rewards' consistency with the labelled ordering

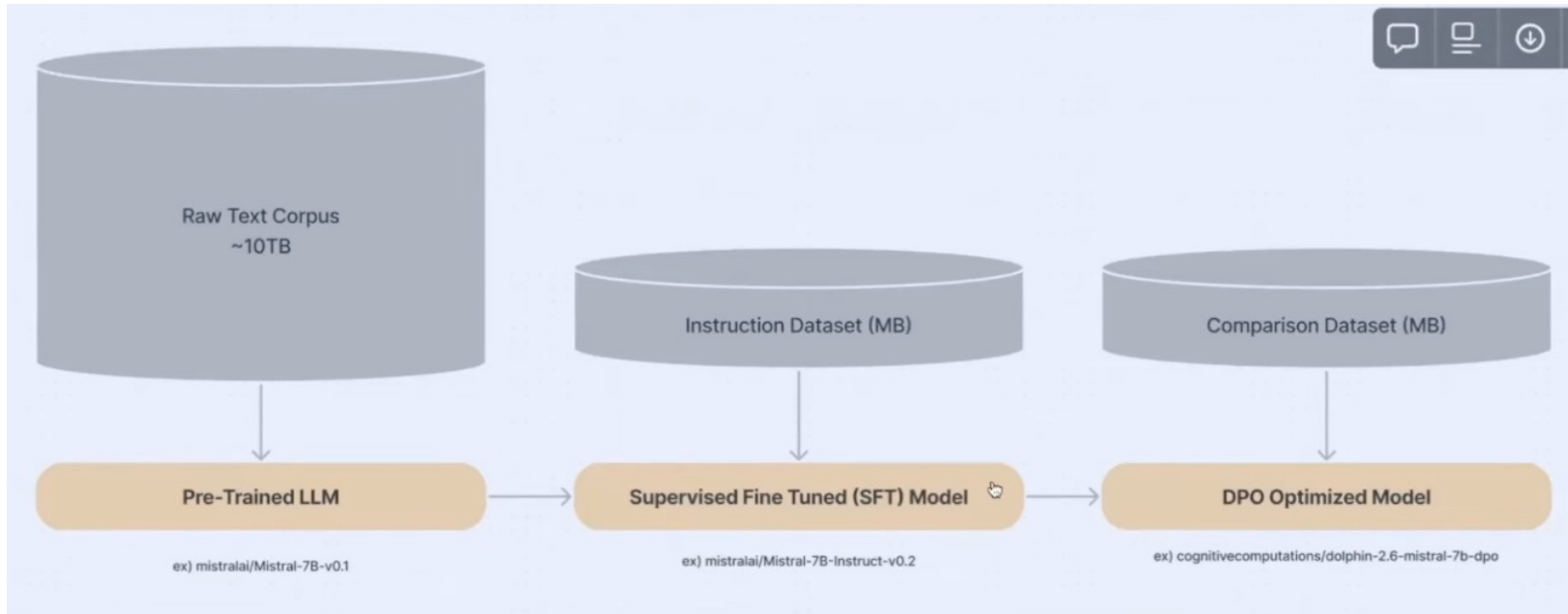
3(b). Reinforcement Learning

Dataset	Prompts, 10-100K prompts prepared by contractors, high quality
Task	Reinforcement learning: generate response that maximize the reward
Output	RL Model, for example ChatGPT
Notes	Init from SFT model, use Reward Model, 1-100 GPUs, days of training

Why RLHF? It's easier to discriminate than to generate.
But it is tricky to get right.

Alternative to RLHF: Direct Preference Optimization (DPO)

DPO is handy, as it removes the need for Reward Model.



Magic: replace RLHF with a “simple” classification loss.

$$\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right].$$