北京大学
PEKING UNIVERSITY

A Brief Introduction to LLM

Kun Yuan

Center for Machine Learning Research @ Peking University

Contents

北京大学 PEKING UNIVERSITY

This lecture is summarized from two wonderful talks [1,2] by Andrej Karpathy

[1] State of GPT

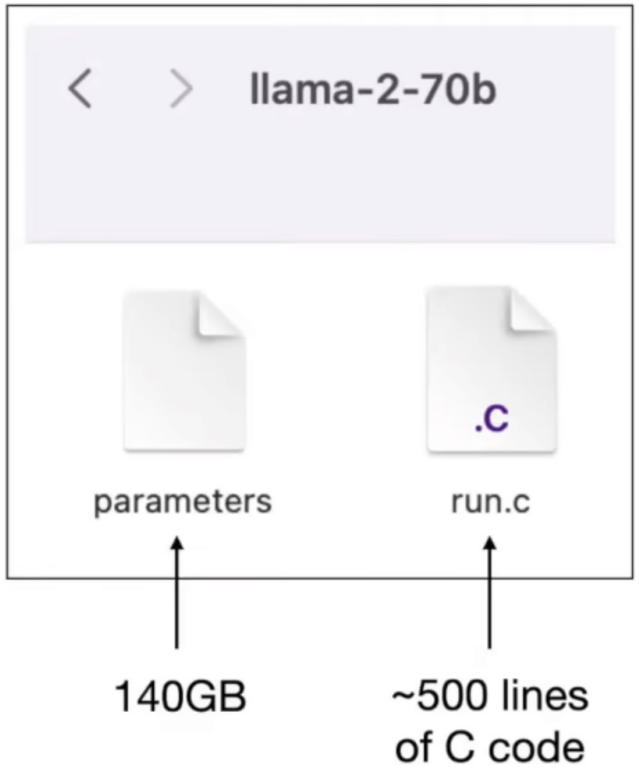
[2] The busy person’s intro to LLMs

北京大学
PEKING UNIVERSITY

PART 01

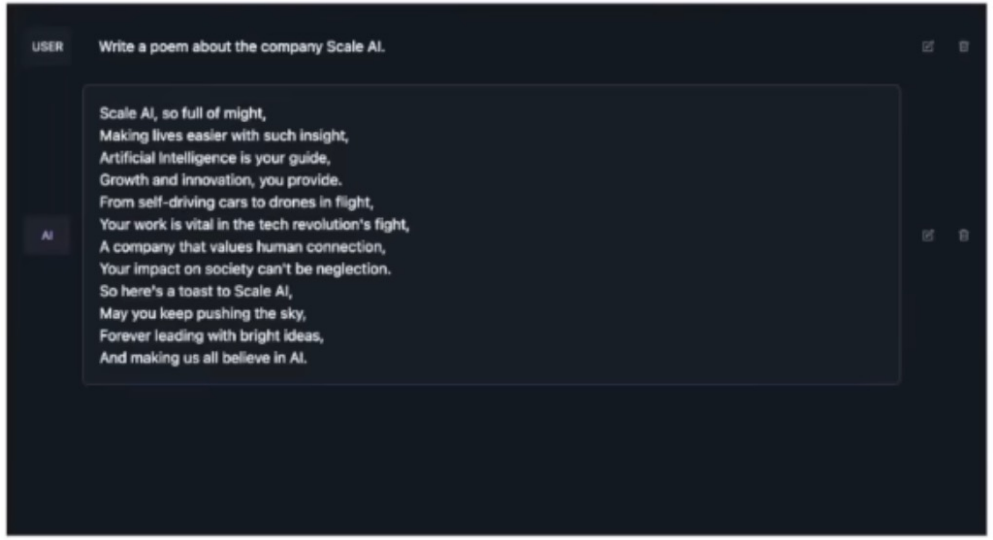
Large language model (LLM)

Large language model



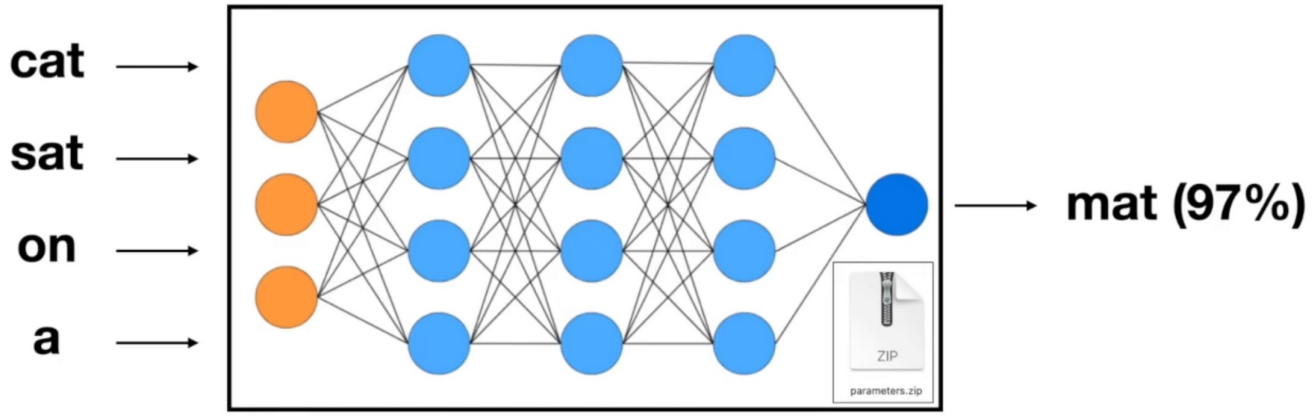
- Meta Llama 3 is a powerful open-source LLM
- Weights, architectures, and the paper were all released by Meta
- Neural network parameters + the code to run them; that's all you need
- No need to access your WIFI. Just one laptop

Large language model



What is the model parameter?

- LLM can be regarded as a magic function that maps the context to the next word
- Model parameter parameterize the magic function to a series of matrix-matrix(vector) products



$$f(\text{“cat sit on a”};\theta) = \text{“mat”}$$

- Given the model parameter θ, LLM can predict the next word

LLM can generate texts of various styles

北京大学 PEKING UNIVERSITY

```python
import numpy as np

class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        # Initialize weights and biases
        self.weights_input_hidden = np.random.randn(self.input_size, self.hidden_size
        self.bias_input_hidden = np.random.randn(1, self.hidden_size)
        self.weights_hidden_output = np.random.randn(self.hidden_size, self.output_si
        self.bias_hidden_output = np.random.randn(1, self.output_size)

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def forward(self, x):
        # Input layer to hidden layer
        self.hidden_input = np.dot(x, self.weights_input_hidden) + self.bias_input_hi
        self.hidden_output = self.sigmoid(self.hidden_input)
```

code

Field	Information
Title	Introduction to Machine Learning
Author	Andrew Ng
Publisher	O'Reilly Media
Publication Date	January 1, 2020
Pages	350
ISBN	978-1492046533
Language	English
Format	Paperback
Edition	First
Categories	Machine Learning, Artificial Intelligence
Description	This book provides a comprehensive introduction to machine learning concepts and techniques, covering topics such as supervised learning, unsupervised learning, and deep learning. It includes practical examples and exercises to help readers understand and apply the principles discussed.

book information

Kun Yuan is an Assistant Professor at Peking University known for his contributions to the field of distributed machine learning. He is recognized for his research in developing scalable and efficient algorithms for distributed machine learning systems, enabling the processing of large-scale datasets across distributed computing environments.

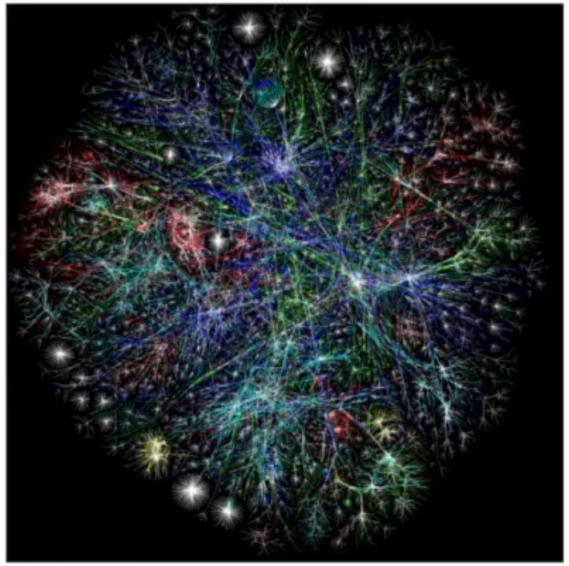
Education and Career:
Kun Yuan obtained his Ph.D. in Computer Science from [University Name], where he focused on distributed systems and machine learning. He later joined Peking University as an Assistant Professor, where he continues to conduct research and teach courses in the areas of distributed computing and machine learning.

Research:
Kun Yuan's research primarily revolves around distributed machine learning, with a particular emphasis on developing algorithms and frameworks that facilitate the training of machine learning models across distributed computing resources. His work often addresses challenges related to scalability, communication efficiency, and fault tolerance in distributed learning systems.

Contributions:
Yuan's research has led to advancements in various aspects of distributed machine learning, including:

wikipedia

How to get the weights? Training the deep neural network



Chunk of the internet,
~10TB of text

6,000 GPUs for 12 days, ~$2M
~1e24 FLOPS

parameters.zip

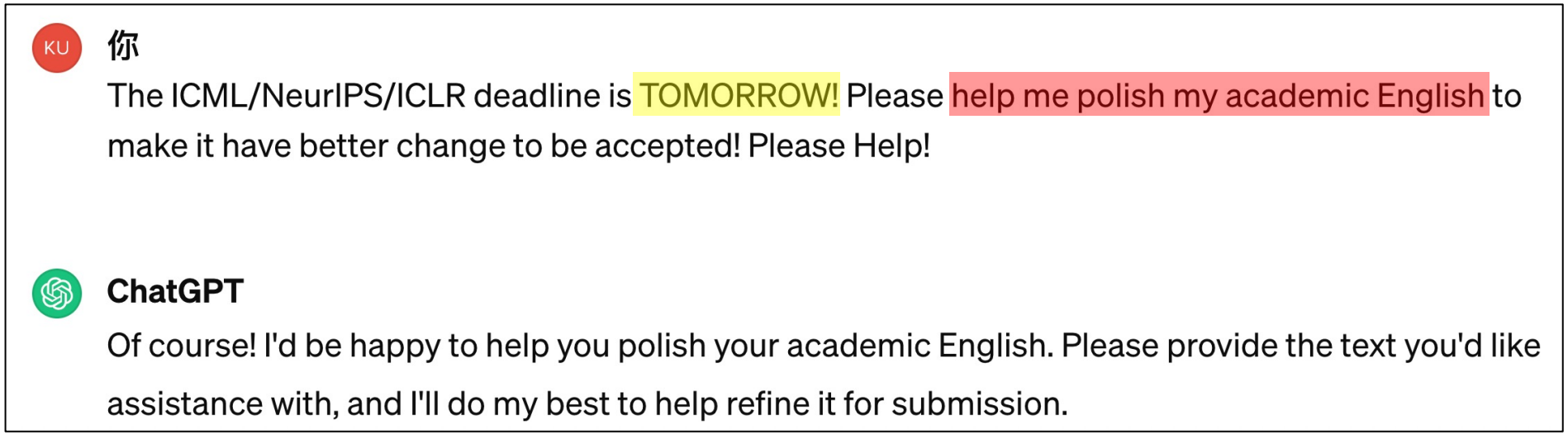
~140GB file

- Use tremendous data and computing resources to get the valuable model parameters
- Very very expensive; update the model weights probably once a year or once a few years

How to make LLM as your personal copilot? PE and finetune

北京大学 PEKING UNIVERSITY

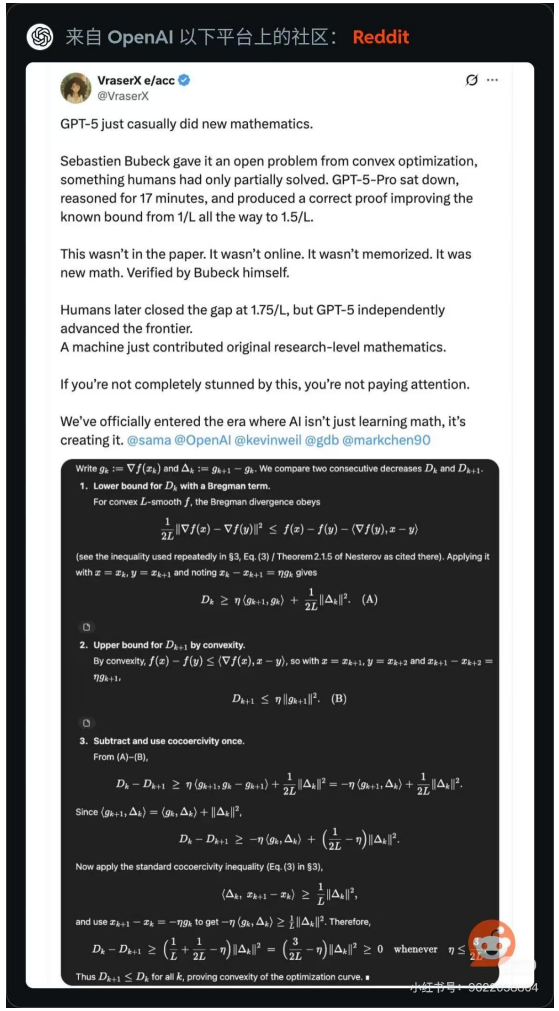
- Over 90% of my interactions with ChatGPT are

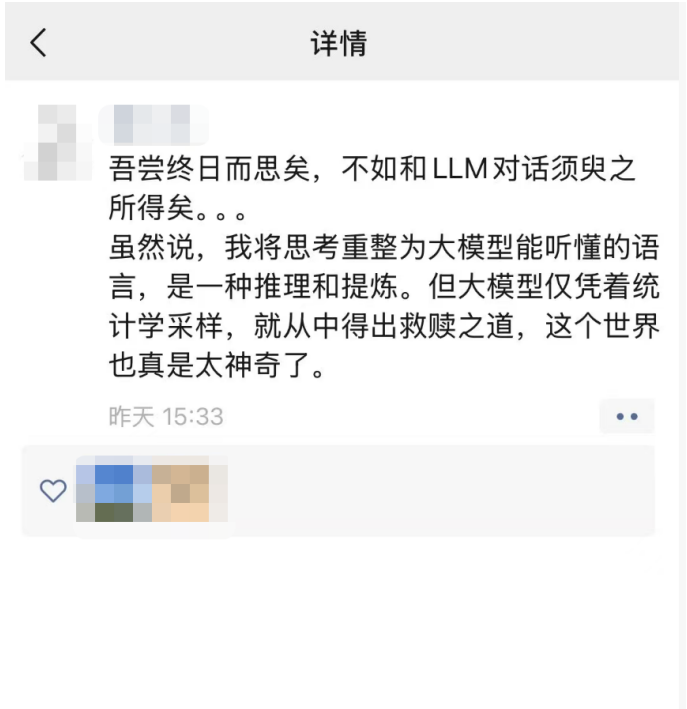


- But we should use LLM more frequently and smartly. It can be your personal copilot
- It is not easy to have your own LLM copilot. You need to know prompt engineering and finetune

How to make LLM as your personal copilot? PE and finetune

ChatGPT can do research



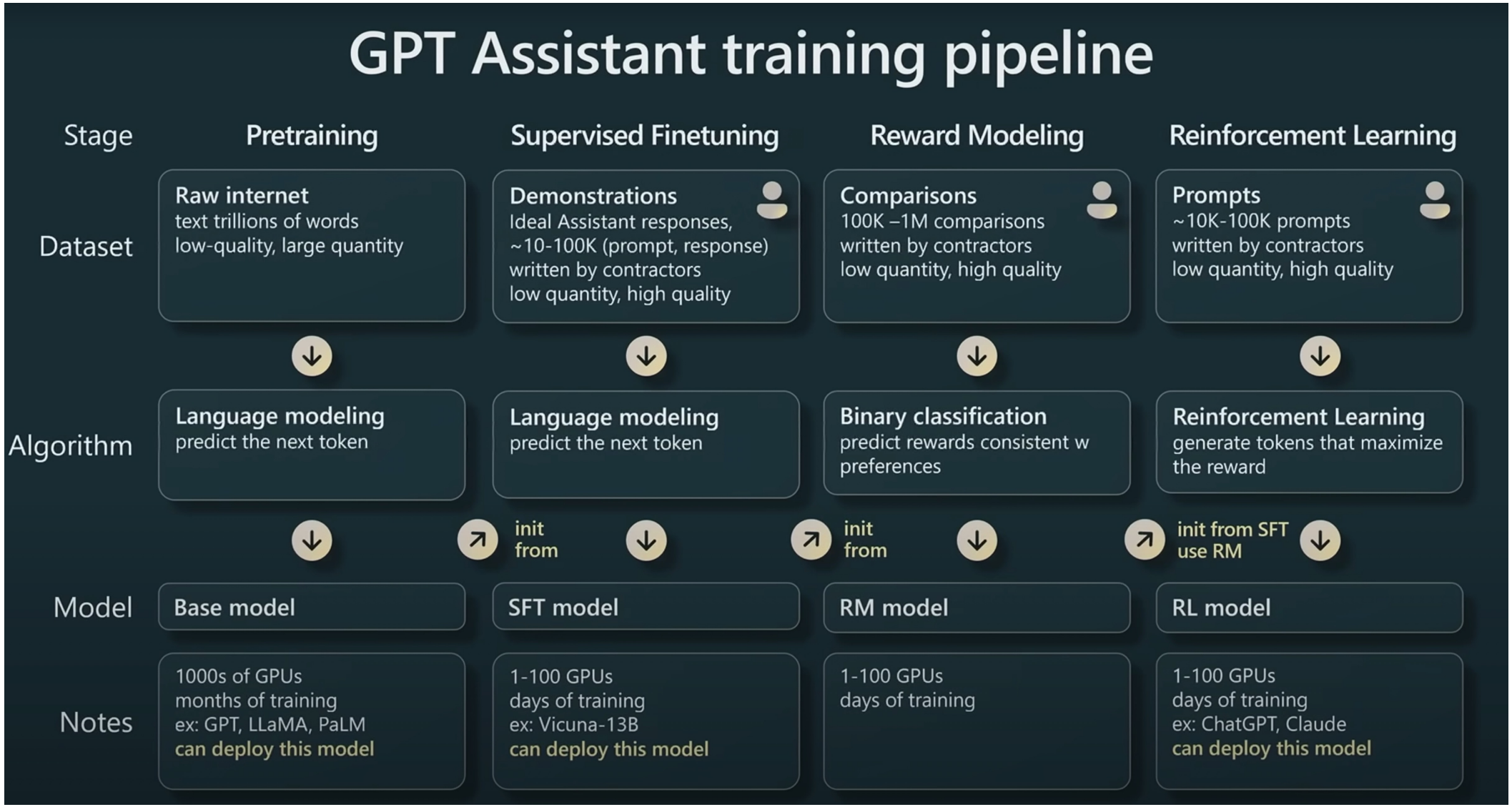


北京大学
PEKING UNIVERSITY

PART 02

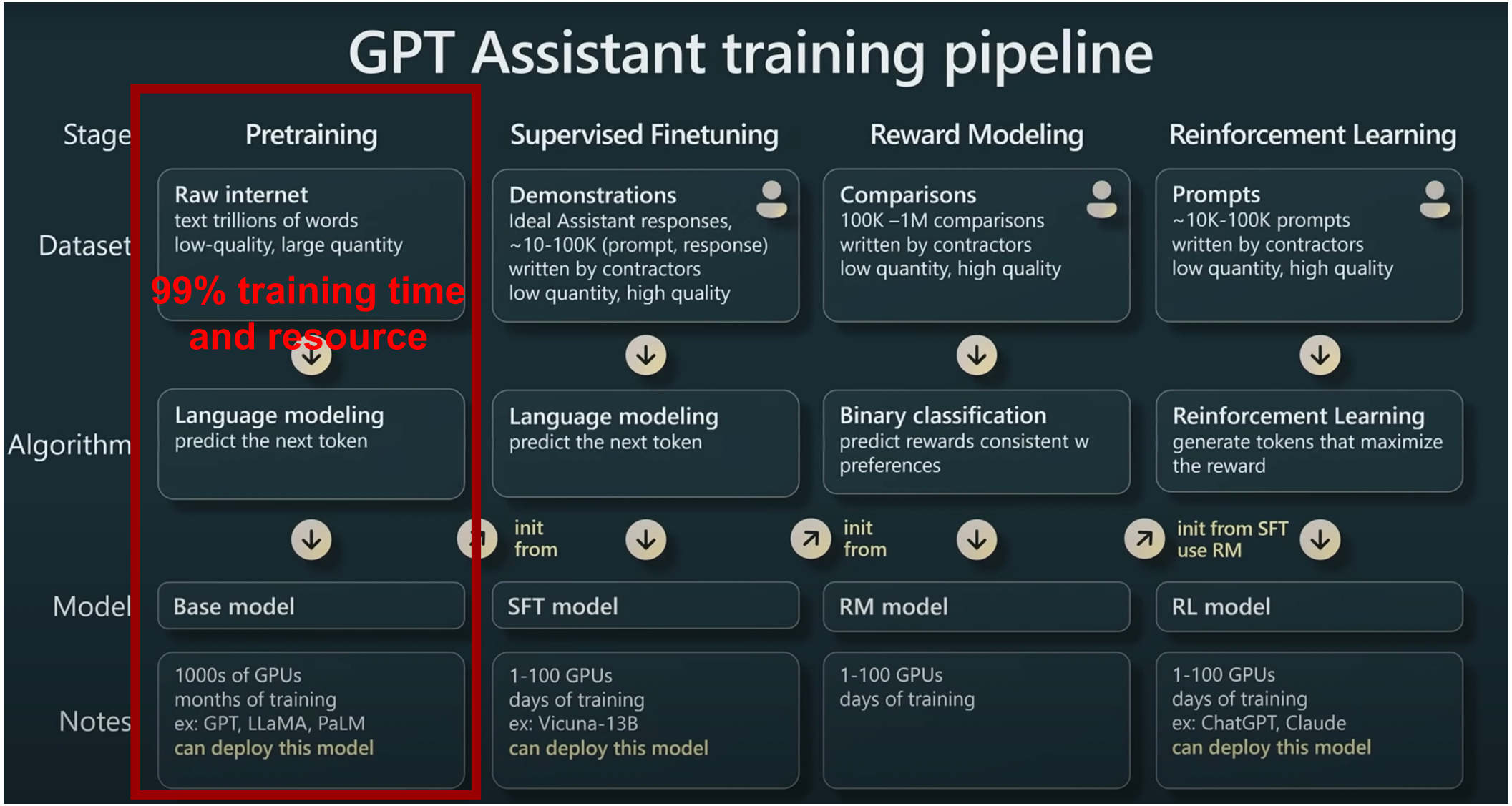
ChatGPT Training Pipeline

ChatGPT training pipeline has 4 stages



Source: Andrej Karpathy, State of GPT

Pretraining



Source: Andrej Karpathy, State of GPT

Pretraining

北京大学 PEKING UNIVERSITY

Data collection

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

Crawled data from websites; in both high quality and low quality

High-quality data

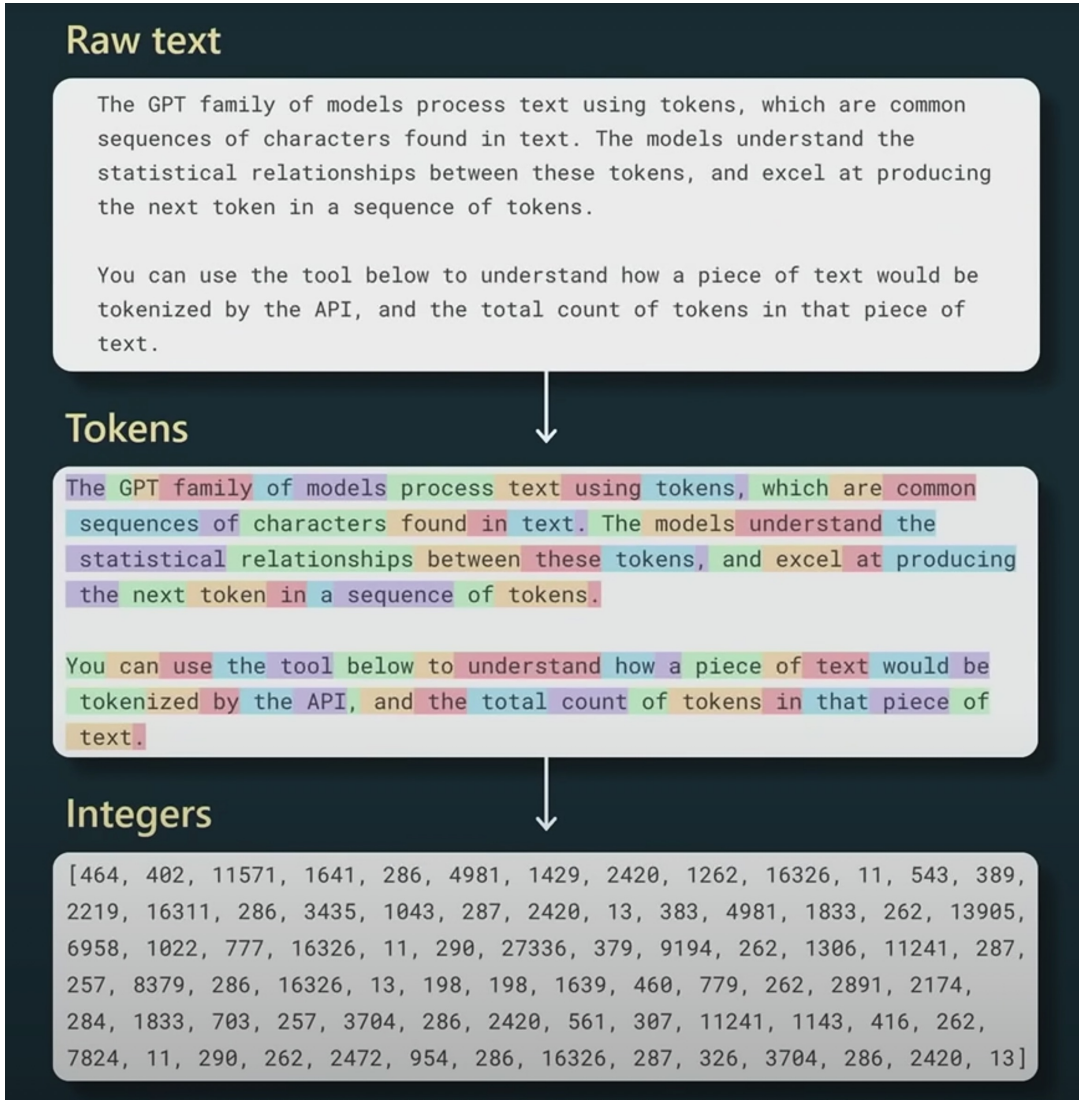
Training data mixture used in lLaMA model

Pretraining

北京大学 PEKING UNIVERSITY

Tokenization (分词)

Transform long texts to lists of integers



Pretraining

北京大学 PEKING UNIVERSITY

Token and vocabulary

Sentence: "The cat sat on the mat. The cat is orange."

Token: ["The", "cat", "sat", "on", "the", "mat", ".", "The", "cat", "is", "orange", "."]

Vocabulary : {"The", "cat", "sat", "on", "the", "mat", ".", "is", "orange"}

Vocabulary is a **set** with each element unique

Pretraining

2 example models

GPT-3 (2020)

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

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Training: (rough order of magnitude to have in mind)

- O(1,000 - 10,000) V100 GPUs
- O(1) month of training
- O(1-10) $M

LLaMA (2023)

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

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Training for 65B model:

- 2,048 A100 GPUs
- 21 days of training
- $5M

Pretraining

While GPT-3 is larger, LLaMa utilizes more tokens. In practice, LLaMA significantly performs better.

We cannot judge the power of one LLM model only by its number of parameters; data also matters

It is still in debate that whether one should increase model size or data size given limited resource budget

Pretraining

The inputs to the Transformer are arrays of shape (B,T)

- B is the batch size (e.g. 4 here)
- T is the maximum context length (e.g. 10 here)

Training sequences are laid out as rows, delimited by special <|endoftext|> tokens

Row 1: Here is an example document 1 showing some tokens.
Row 2: Example document 2<|endoftext|>Example document 3<|endoftext|>Example document
Row 3: This is some random text just for example<|endoftext|>This
Row 4: 1,2,3,4,5

One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

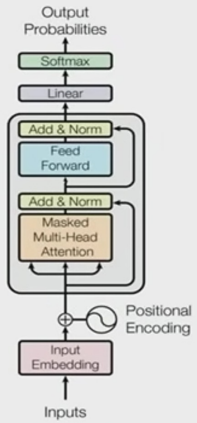
Pretraining

Pretraining

Each cell only "sees" cells in its row, and only cells before it (on the left of it), to predict the next cell (on the right of it)

Green = a random highlighted token
Yellow = its context
Red = its target

50,257 numbers
(probabilities for the next token)
correct index (label): 513



Transformer
(neural network)

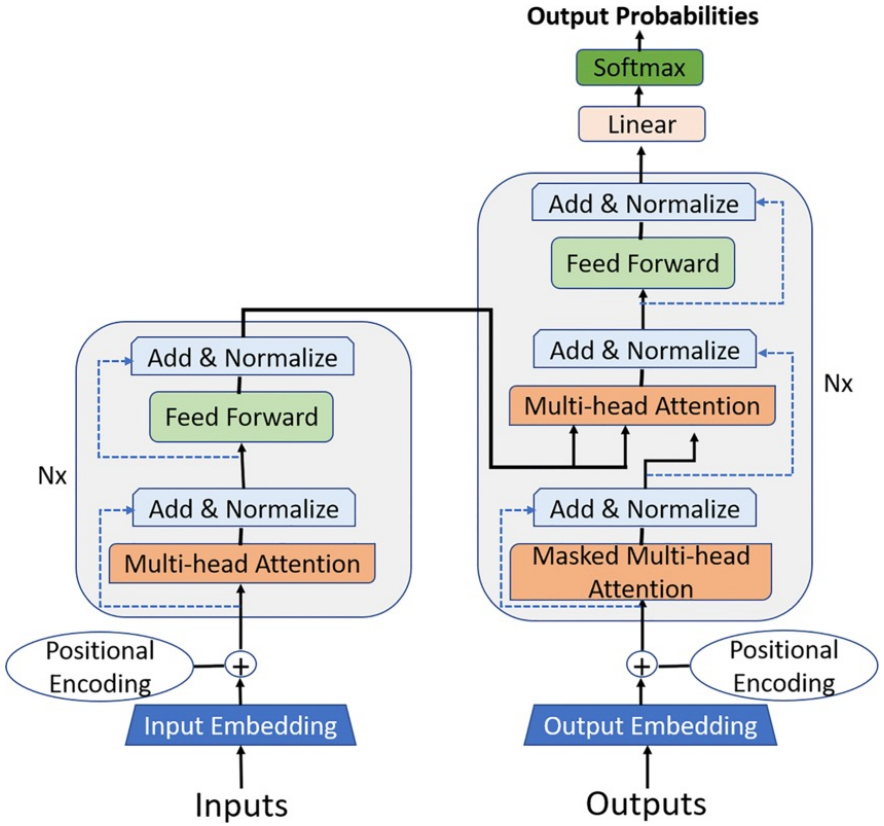
One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Pretraining



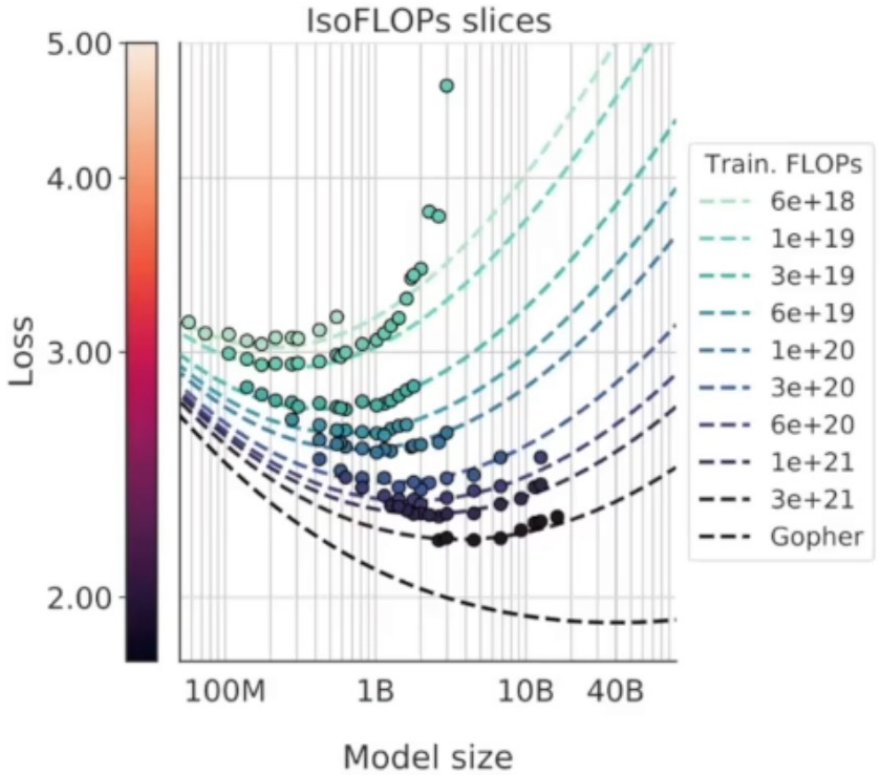
Transformer architecture
(will discuss it in later lectures)

- Effective representation learning
- Long-range dependency with attention
- Parallelizable architecture
- Flexibility and Adaptability

  (In recent popular SORA, Diffusion + transformer is used)

Pretraining

[Training Compute-Optimal Large Language Models]



Amazing representation power

Larger dataset + bigger model + longer training

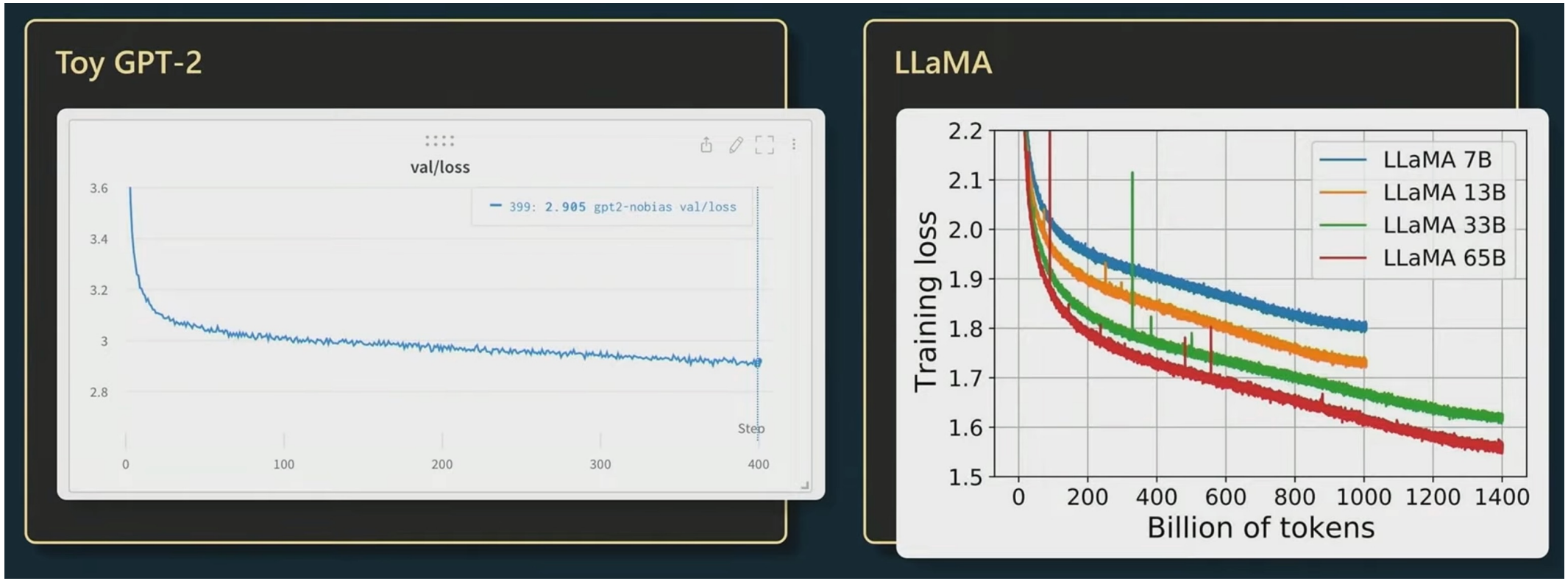
=

better prediction accuracy

A very straightforward way to achieve good LLM.

All you need is **MONEY**!

Pretraining



Larger dataset + bigger model + longer training = better prediction accuracy

Pretraining

北京大学 PEKING UNIVERSITY

Training process

Training data (Shakespeare)

```
First Citizen:
We cannot, sir, we are undone already.

MENENIUS:
I tell you, friends, most charitable care
Have the patricians of you. For your wants,
Your suffering in this dearth, you may as well
Strike at the heaven with your staves as lift them
Against the Roman state, whose course will on
The way it takes, cracking ten thousand curbs
Of more strong link asunder than can ever
Appear in your impediment. For the dearth,
The gods, not the patricians, make it, and
Your knees to them, not arms, must help. Alack,
You are transported by calamity
Thither where more attends you, and you slander
The helms o' the state, who care for you like fathers,
When you curse them as enemies.
```

Samples at initialization

z'v}yy_RMV(7ea
AOCEi2tfEi lermh`
`88]gLNSSx|6Mj"i1wdcf,WezVII<4x?OBhS7D-}.8wCkGFgB(kC-
h'Ywa.QhjPo,3C.dA!3;_]!AKa.e0Ml Iz(DqAfE8.}nm32<Z2ma1,6DAp
xOrA"jA[V;yhD]<g?BjKXbuptt|W:RT8,ti"(h8J"b"](ZPv3uExA.2r<&;wI?
`mnGs]MG8saNr3"u7tAftthhQBt`GEu66DxN'[[`LU!fUXhy!LI2DjK a
b("8GL``Z66Dhv0,ooqv.
5nmUeh _'j}jjjW33EClY(5l
0vwdE;_Ze`veBbUv<y'TTBk(m]67q`1N`pd|EobQQ]RtKDXii0Y,LwOZ8d'y1)u
7d|N''CIE2y4hS"Ml0od3vtDVV<P``J1ONNn]Y4S<`Q}I2e9d2r8_
ccw[h'9TKFz]8IIDBlh'0y91i?<SKKL'sBv}v

[GPT from scratch, NYT, 2023]

Samples after 250 iterations of training

ONom hende beer'TIAFRO.
Rome thecoramerert BENRABENBUR. Nore se. he llod hears hy pid gof
wiere the the paron deread boan: ins wtherk hof at f o otherira coust Soot,
Hyou seealler sheron mer w f shathe thatchie anden wer by he thew bat

Samples after 500 iterations of training

For but te aser if the coouldlavilcoon Creater?
RANTEBR. In fease. Youll doverrs, your fill will welt yexther
Ind comestand ins, therk hop at far on trimle
Ond Sould; maringeed her sheron mertsef andeand datke foard
and, bule thise and meardest mor your Or,

Samples after 5,000 iterations of training

Hor. I have been me, thereof my life, and he concludes him.
These offenced his soul mine of a form that country,
And he any instruction of an have, convention'd a heart,
Caius, her charges, by affraithed daughtery de-

Samples after 30,000 iterations of training

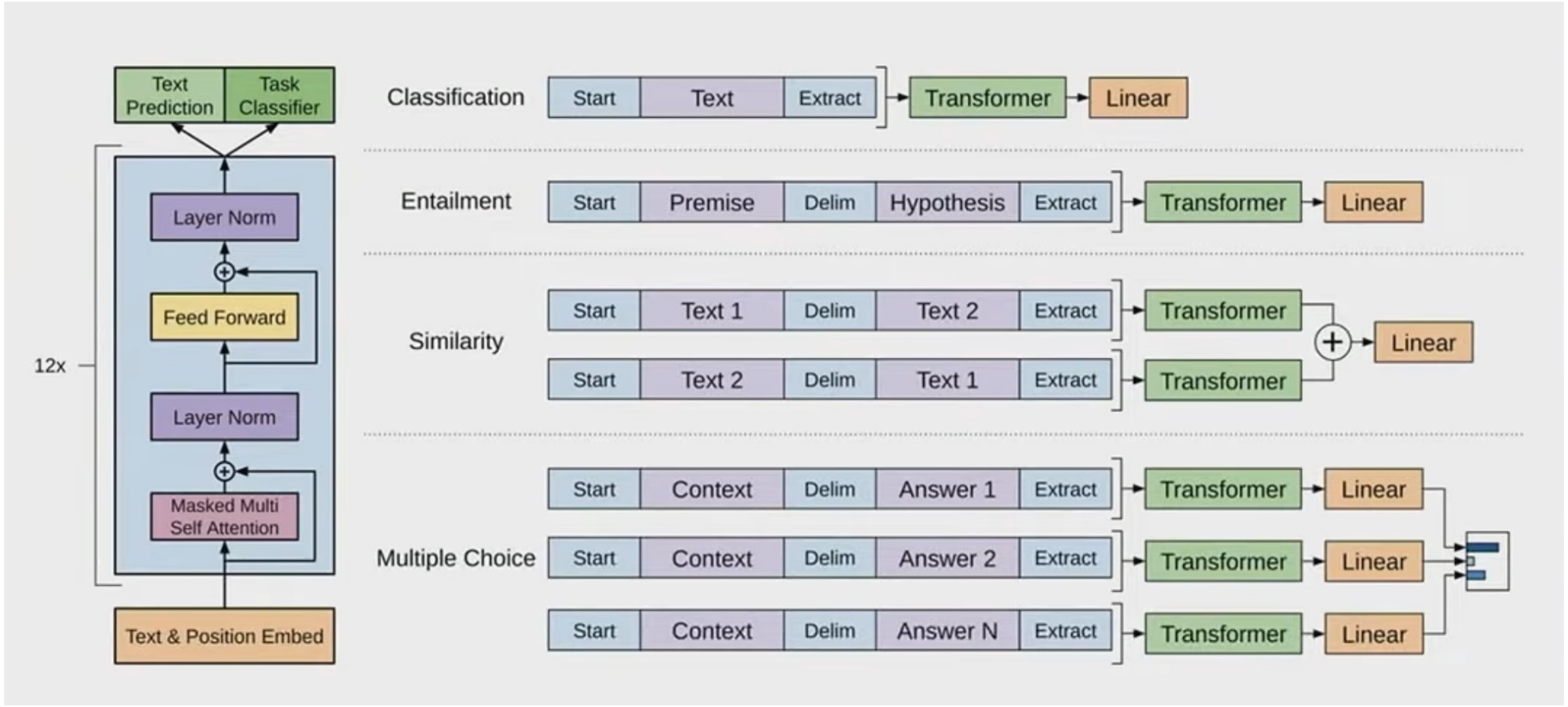
Of gold that breeds forth thou must like the stars,
But they are sent soldiers, her window in their states,
And speak withal: if the Lord of Hereford,
With court to this person all the King mercy

Pretraining

北京大学 PEKING UNIVERSITY

- Pretraining a base model is extremely expensive
- Several effective pretraining techniques:
  - 3D parallelism: data/model/tensor parallelism
  - Memory-efficient optimizers
  - Large-batch training
  - Mixed-precision training
- Will discuss them in later lectures

Pretrained model provides strong transfer learning capabilities



Pretrained base model performs well after finetuning

Pretrained model provides strong transfer learning capabilities

北京大学 PEKING UNIVERSITY

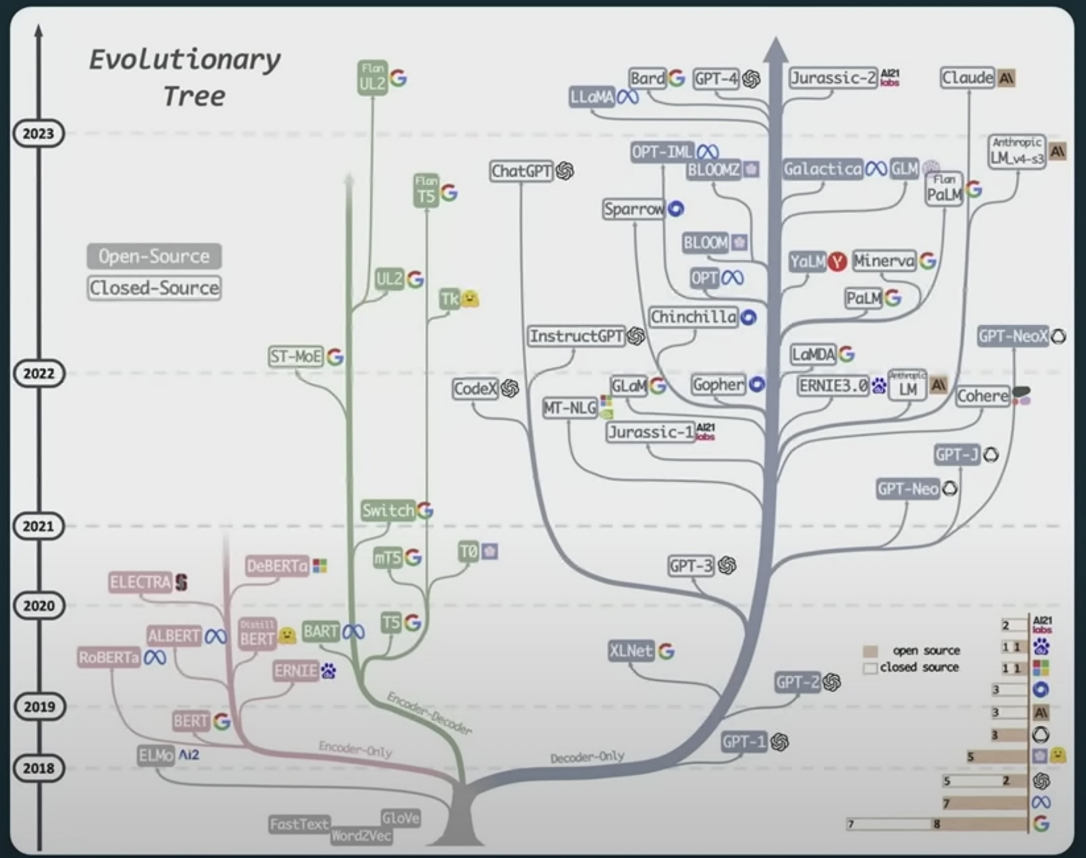
- Pretraining + finetuning/prompting reshapes the AI industry.

- Pretrained base model only needs a small amount of data to be adapted to the down-stream applications.

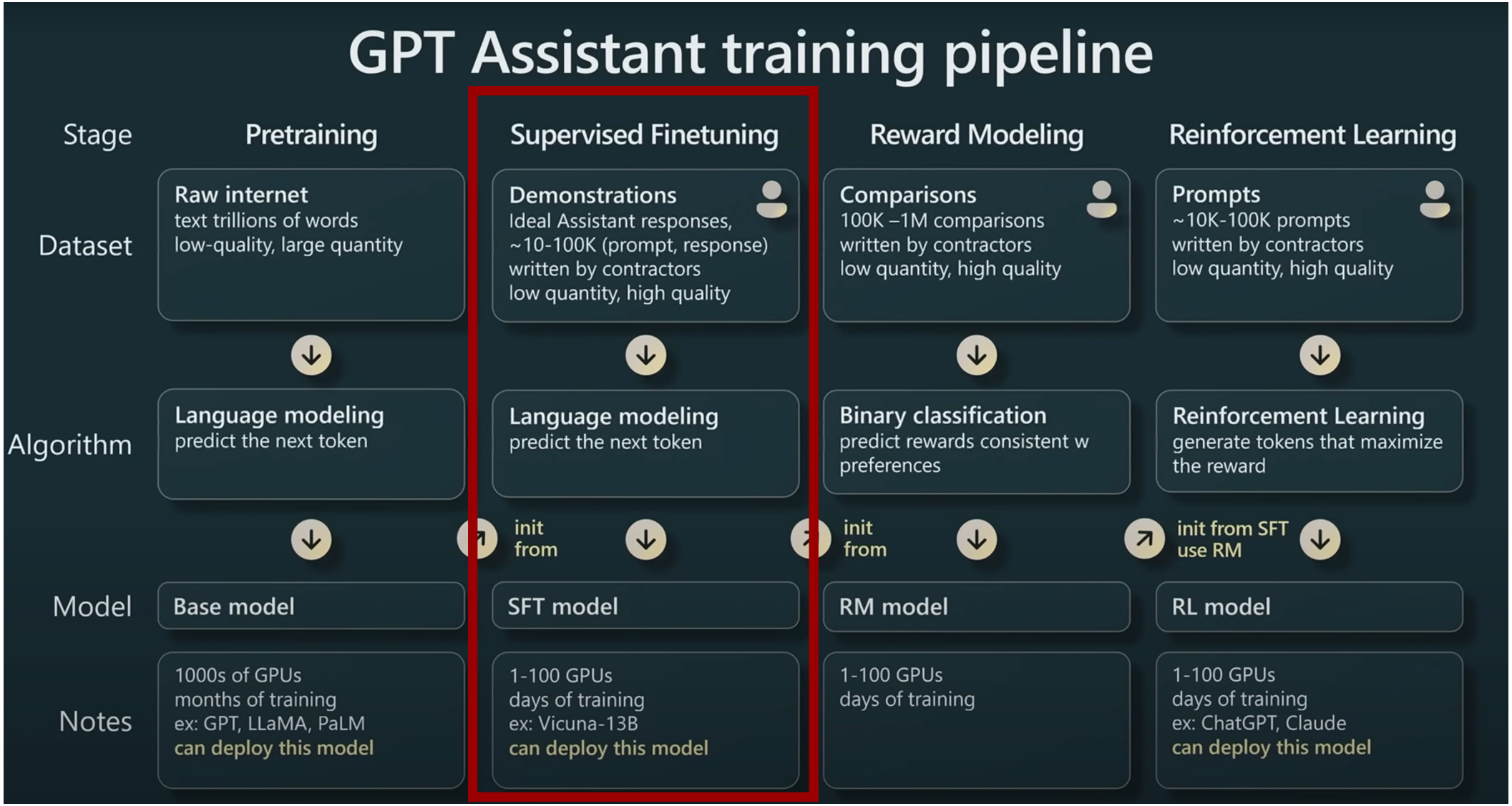
- The cost to deploy AI to down-stream applications decreases significantly
  - Achieve powerful base models from OpenAI/Google/Meta/GitHub
  - Collect a small number of downstream data and use it to finetune the base model / Or prompt LLM
  - No need for expensive investment of money and talents

Base models in the wild

- GPT **Improving Language Understanding by Generative Pre-Training**. 2018. Paper
- GPT-2 **Language Models are Unsupervised Multitask Learners**. 2018. Paper
- GPT-3 **"Language Models are Few-Shot Learners"**. NeurIPS 2020. Paper
- OPT **"OPT: Open Pre-trained Transformer Language Models"**. 2022. Paper
- PaLM **"PaLM: Scaling Language Modeling with Pathways"**. *Aakanksha Chowdhery et al.* arXiv 2022. Paper
- BLOOM **"BLOOM: A 176B-Parameter Open-Access Multilingual Language Model"**. 2022. Paper
- MT-NLG **"Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model"**. 2021. Paper
- GLaM **"GLaM: Efficient Scaling of Language Models with Mixture-of-Experts"**. ICML 2022. Paper
- Gopher **"Scaling Language Models: Methods, Analysis & Insights from Training Gopher"**. 2021. Paper
- chinchilla **"Training Compute-Optimal Large Language Models"**. 2022. Paper
- LaMDA **"LaMDA: Language Models for Dialog Applications"**. 2021. Paper
- LLaMA **"LLaMA: Open and Efficient Foundation Language Models"**. 2023. Paper
- GPT-4 **"GPT-4 Technical Report"**. 2023. Paper
- BloombergGPT **BloombergGPT: A Large Language Model for Finance**, 2023, Paper
- GPT-NeoX-20B: **"GPT-NeoX-20B: An Open-Source Autoregressive Language Model"**. 2022. Paper

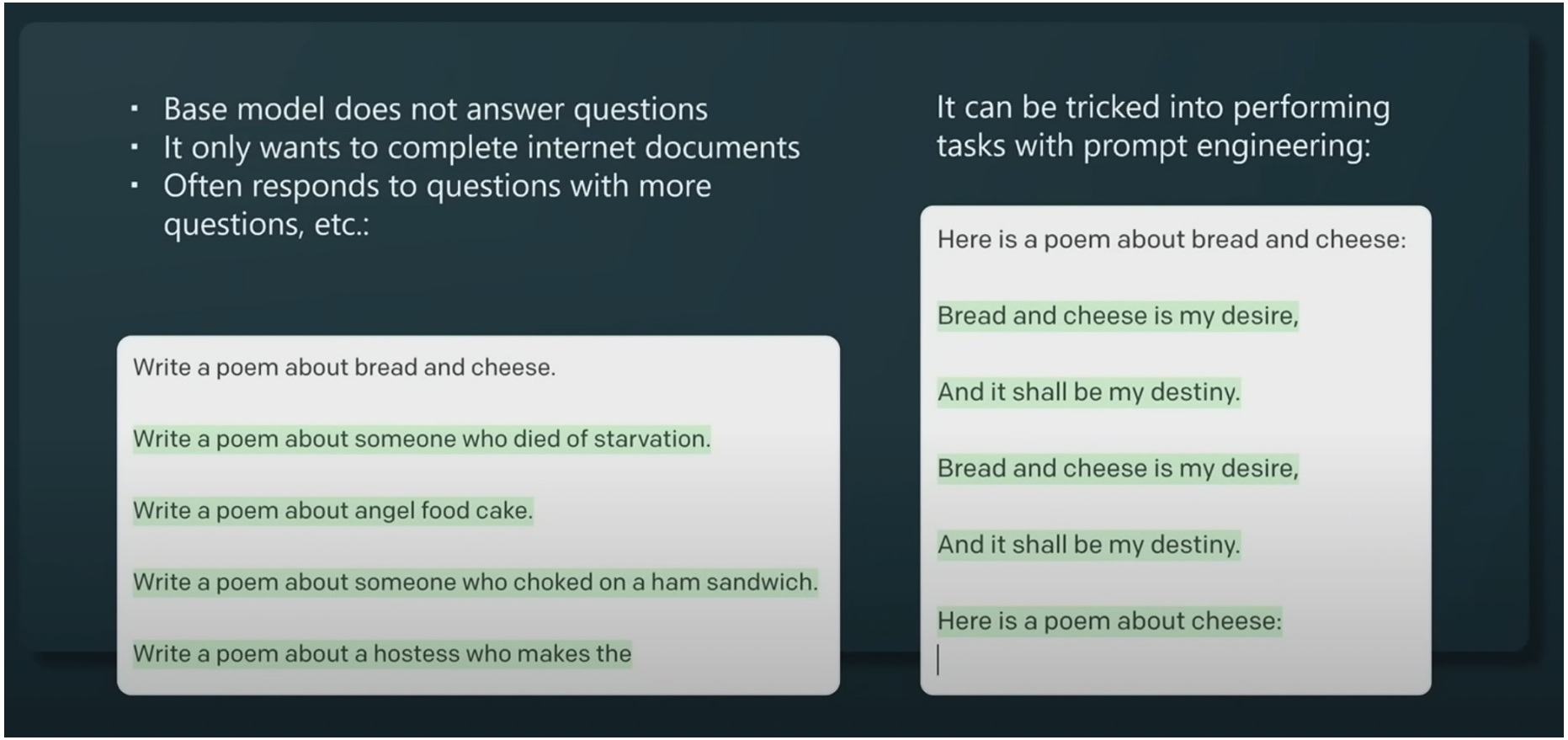


Supervised Finetuning



Supervised Finetuning

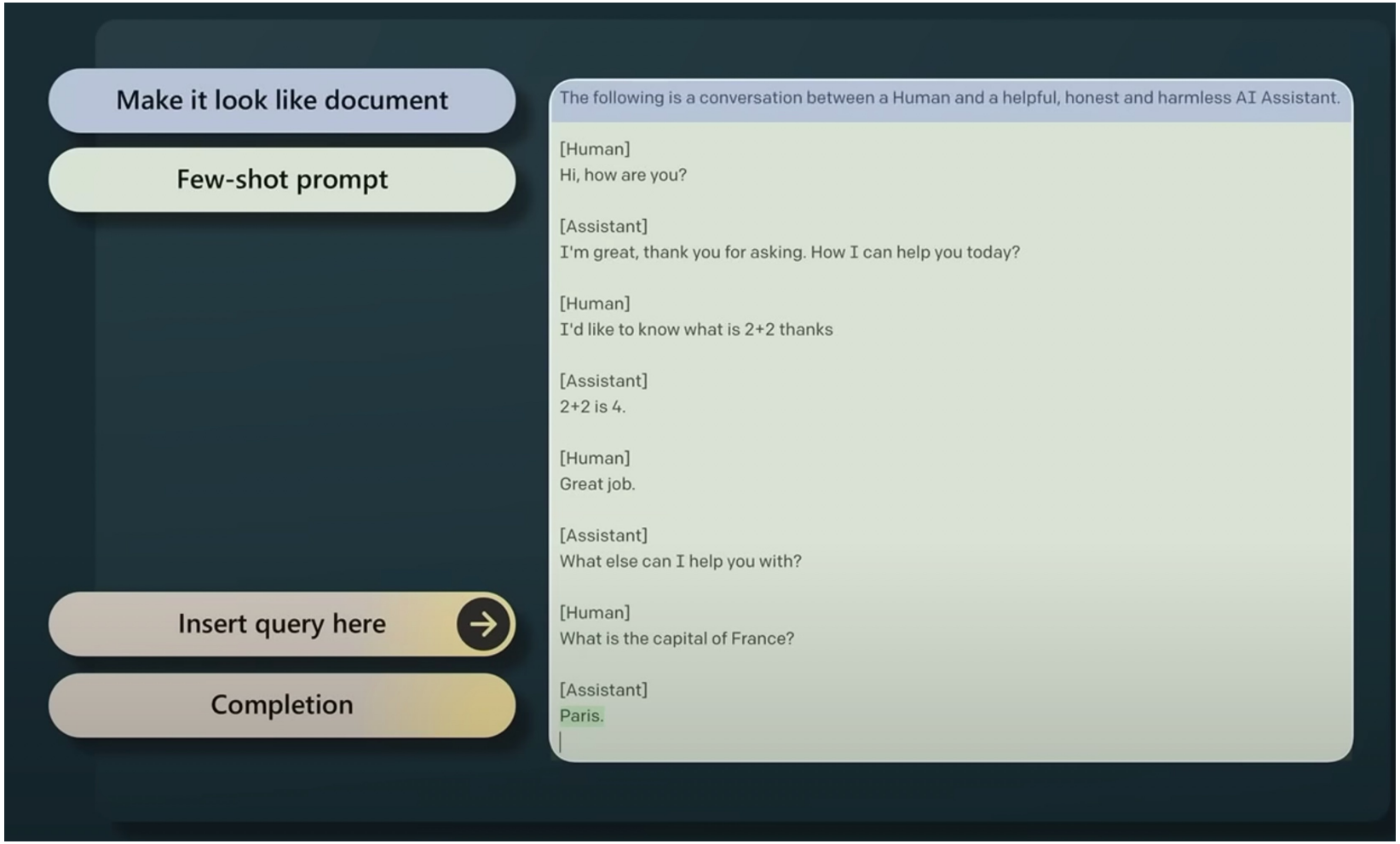
Base models cannot be deployed directly. It is still far away from being a smart assistant



Supervised Finetuning

Base models can be tricked into being AI assistants with prompting

We need to finetune the base model to make it chat like humans



Supervised Finetuning

北京大学 PEKING UNIVERSITY

Ask human contractors to respond to prompts and generate high-quality, **helpful, truthful, and harmless** responses

Collect 10,000+ high-quality human-generated responses

Finetune base models with these high-quality data

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.

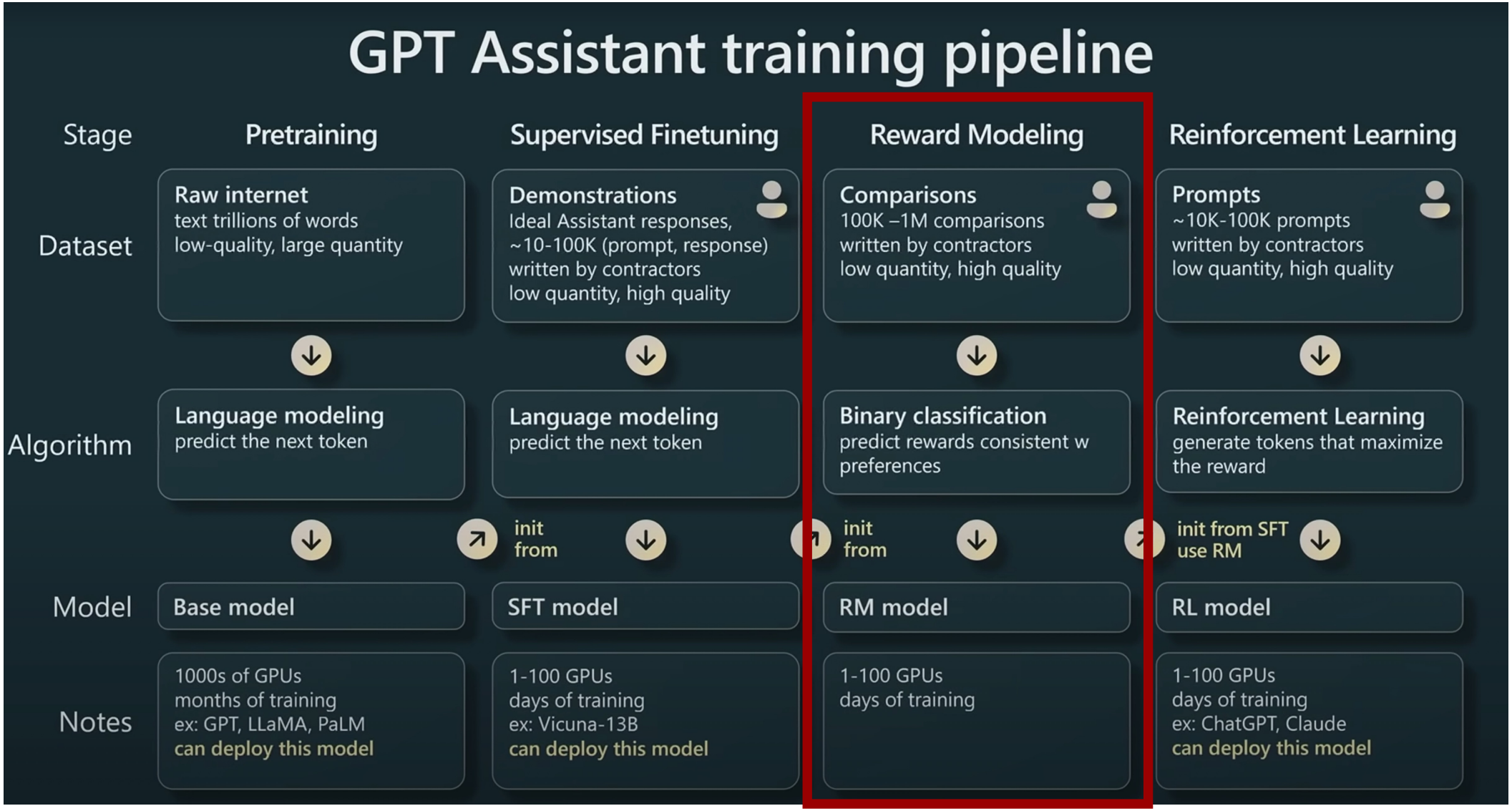
[InstructGPT]

Supervised Finetuning

北京大学 PEKING UNIVERSITY

- Dataset: 10~100K human-generated data pairs {(prompt, response)}
- Training: repeat what we did in the “Pretraining” stage
- After supervised finetuning stage, base models can chat like humans
- 1-100 GPUs; days of training; but can still be very expensive due to human-generated data
- To save money, some (or most) models use ChatGPT-generated data to finetune

Reward modeling



Reward modeling

北京大学 PEKING UNIVERSITY

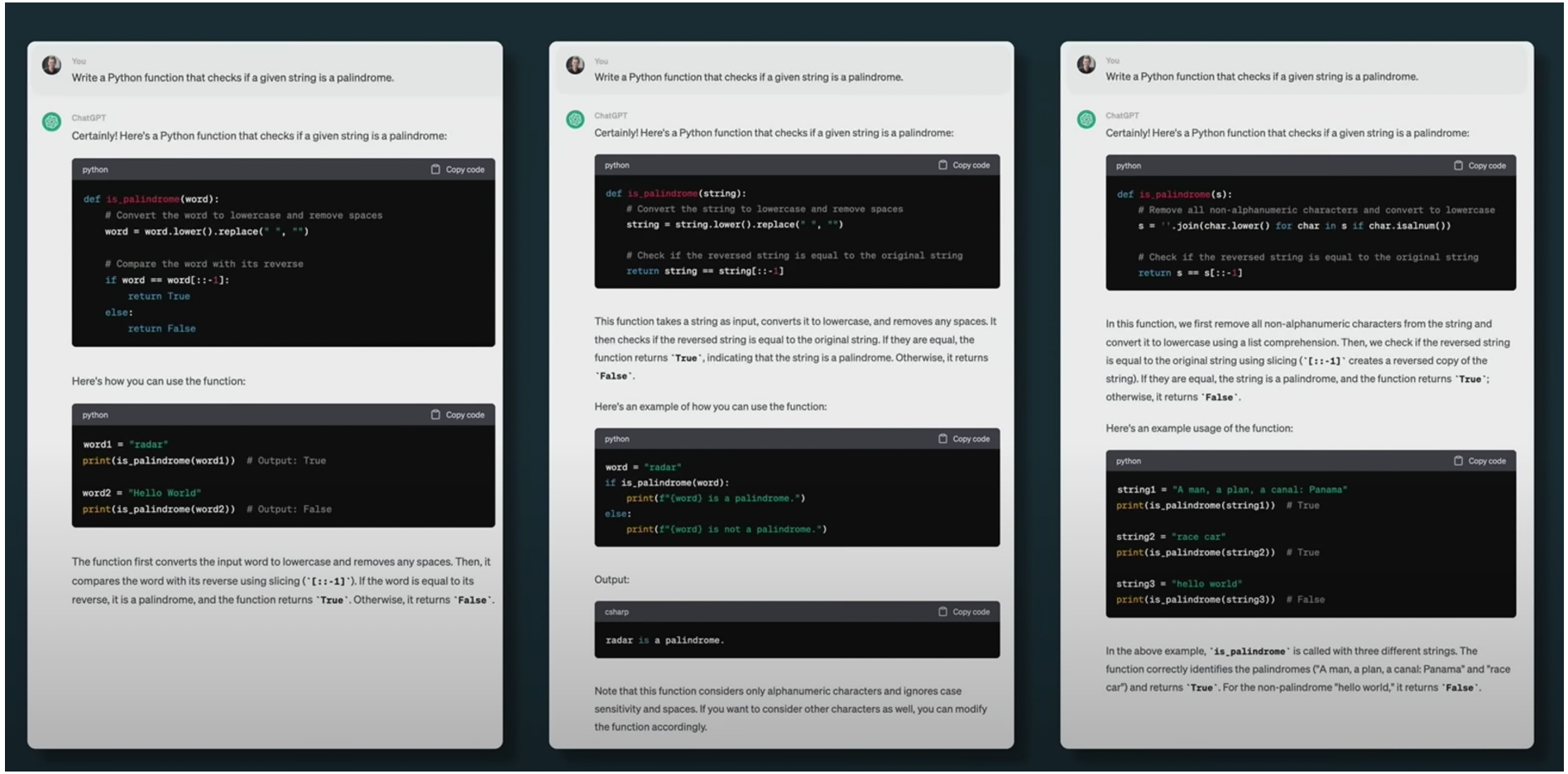
- SFT model performs like an “assistant”, but still not good enough.
- To further improve it, one can ask human contractors to generate more data; effective but expensive
- Another way is to let the model learn what response is good, and how to generate good response
- Reward model will enable GPT to judge whether a certain response is good or not
- Reward model will be used in the Reinforcement learning stage to reinforce good response

Reward modeling

北京大学 PEKING UNIVERSITY

SFT model generates different responses to the same prompt

Dataset



You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```python
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:

```python
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`.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```python
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:

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

Output:

```csharp
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.

You
Write a Python function that checks if a given string is a palindrome.

ChatGPT
Certainly! Here's a Python function that checks if a given string is a palindrome:

```python
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:

```python
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`.

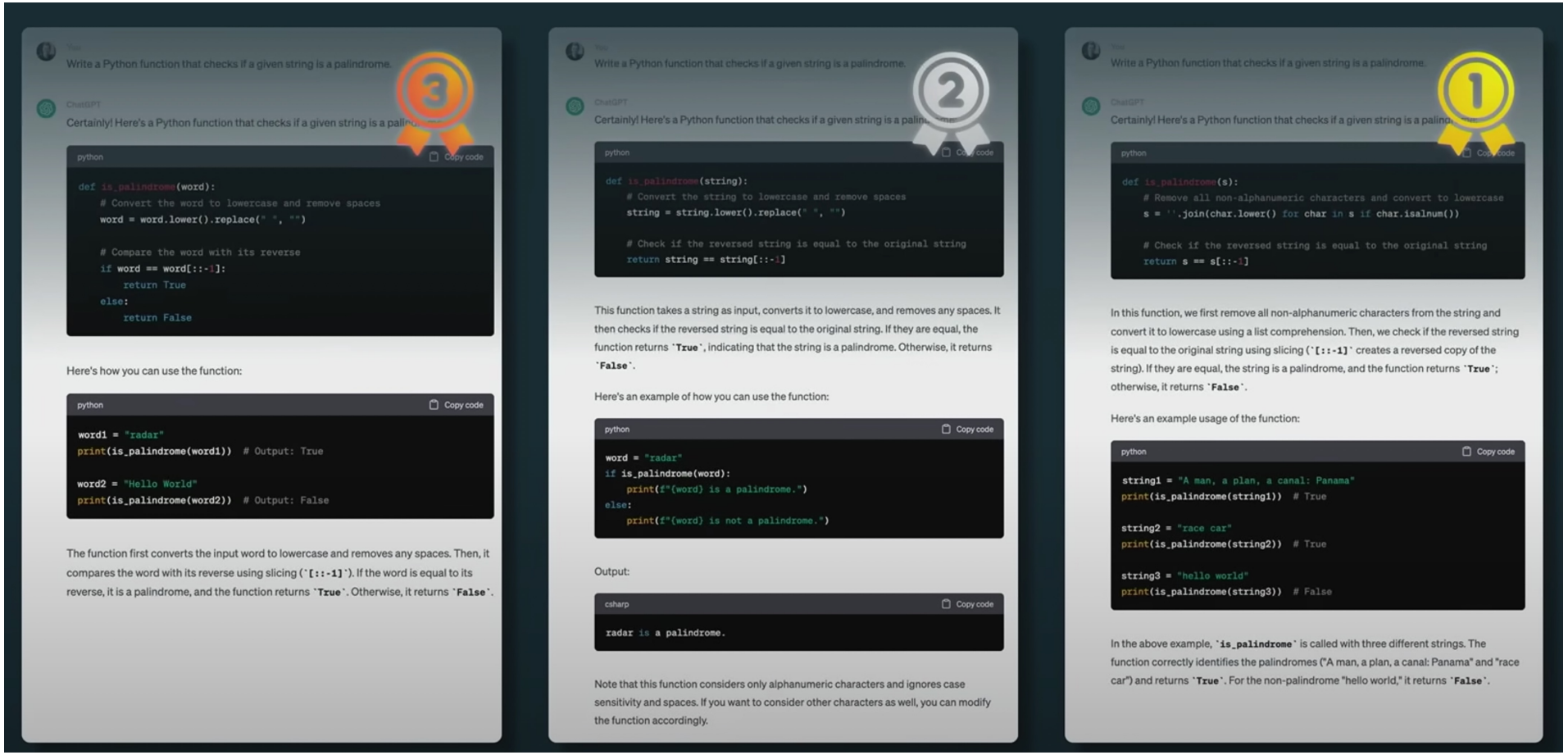
Reward modeling

北京大学 PEKING UNIVERSITY

SFT model generates different responses to the same prompt

Ask contractors to rank the responses; much cheaper

Dataset



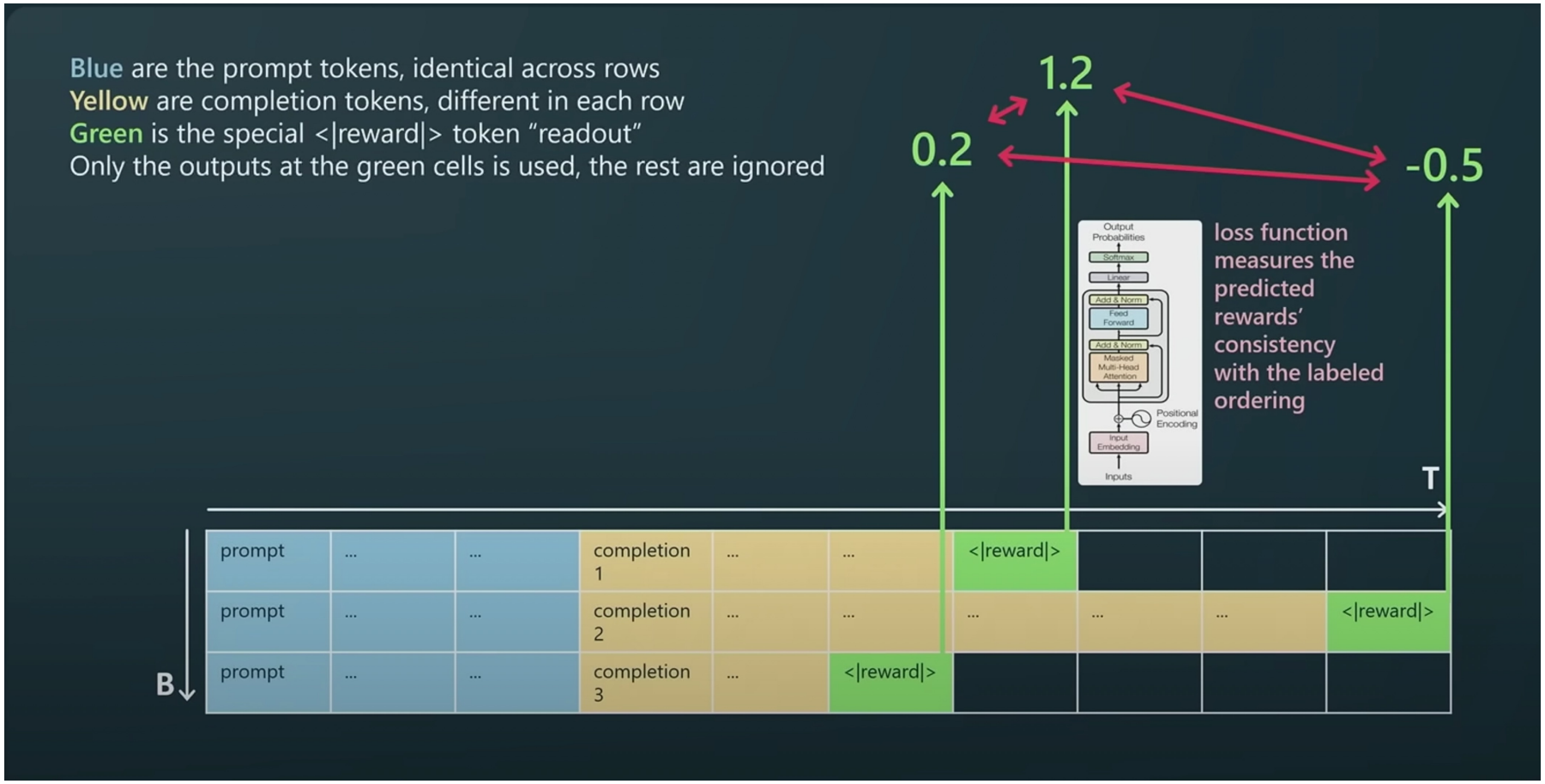
Reward modeling

SFT model generates different responses to the same prompt

Ask contractors to rank the responses; much cheaper

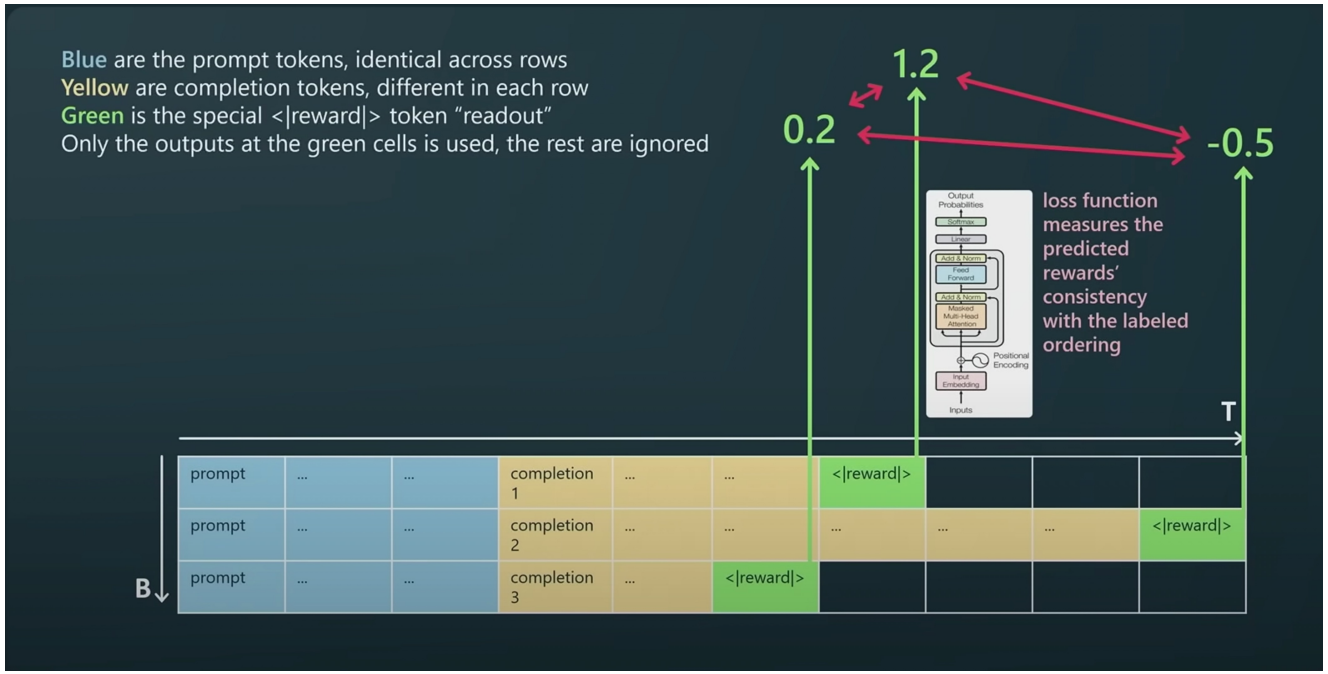
Dataset: {(prompt, response, reward)}

Dataset

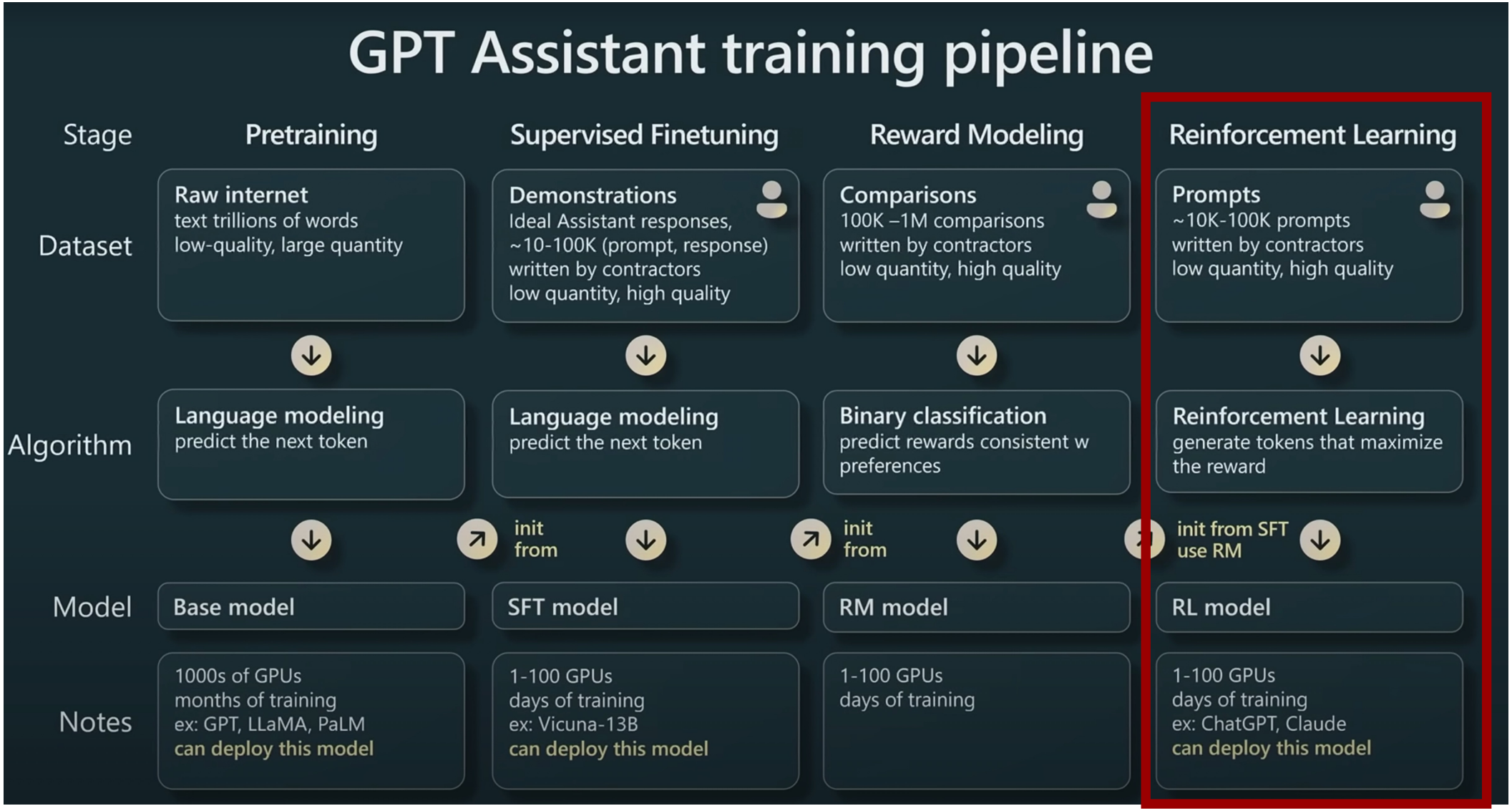


Reward modeling

- Given a prompt, SFT model generates several responses, and then makes a reward prediction (green).
- This reward will be supervised by ground-truth reward.
- After training, we achieve a RW model that can predict the reward after its generated response.

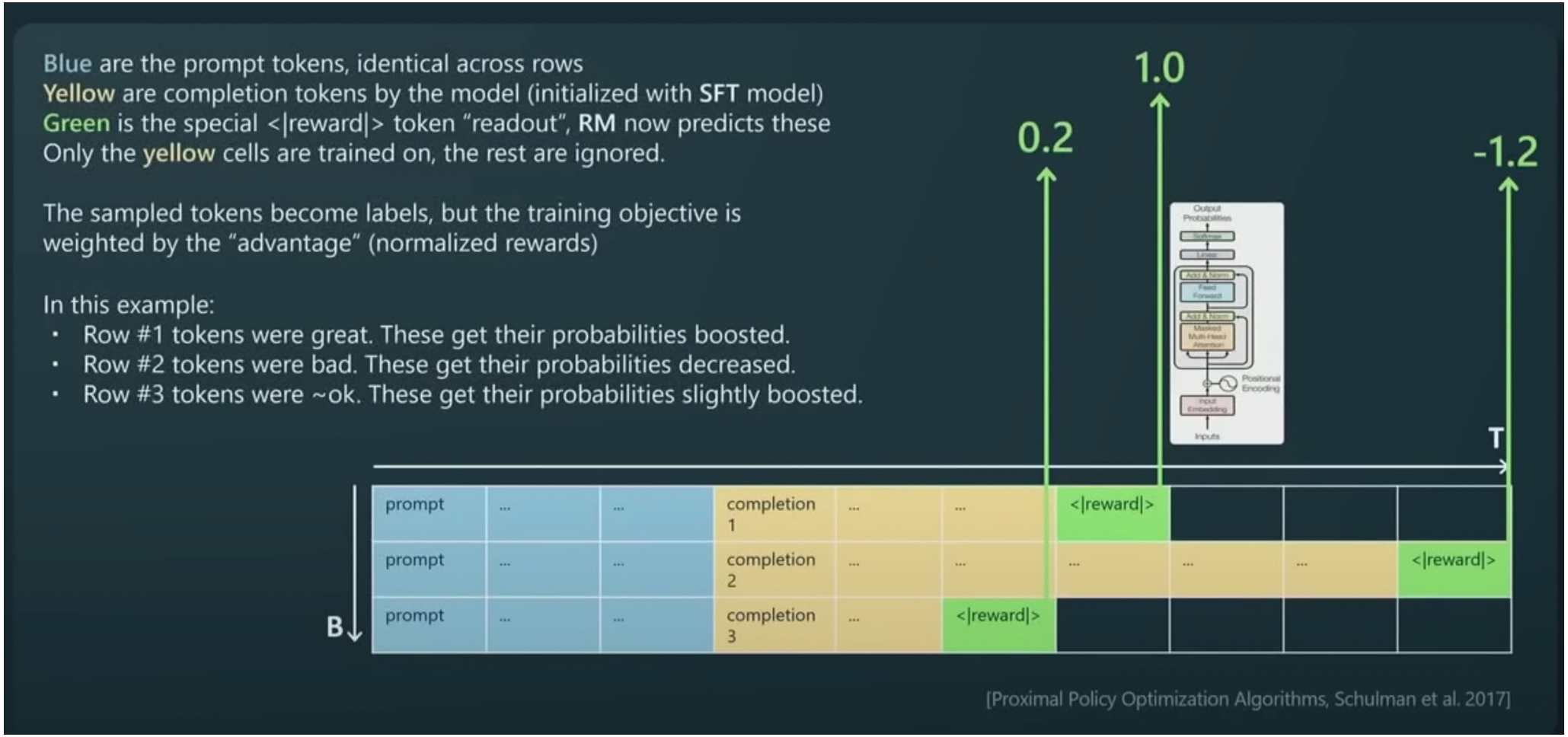


Reinforcement learning

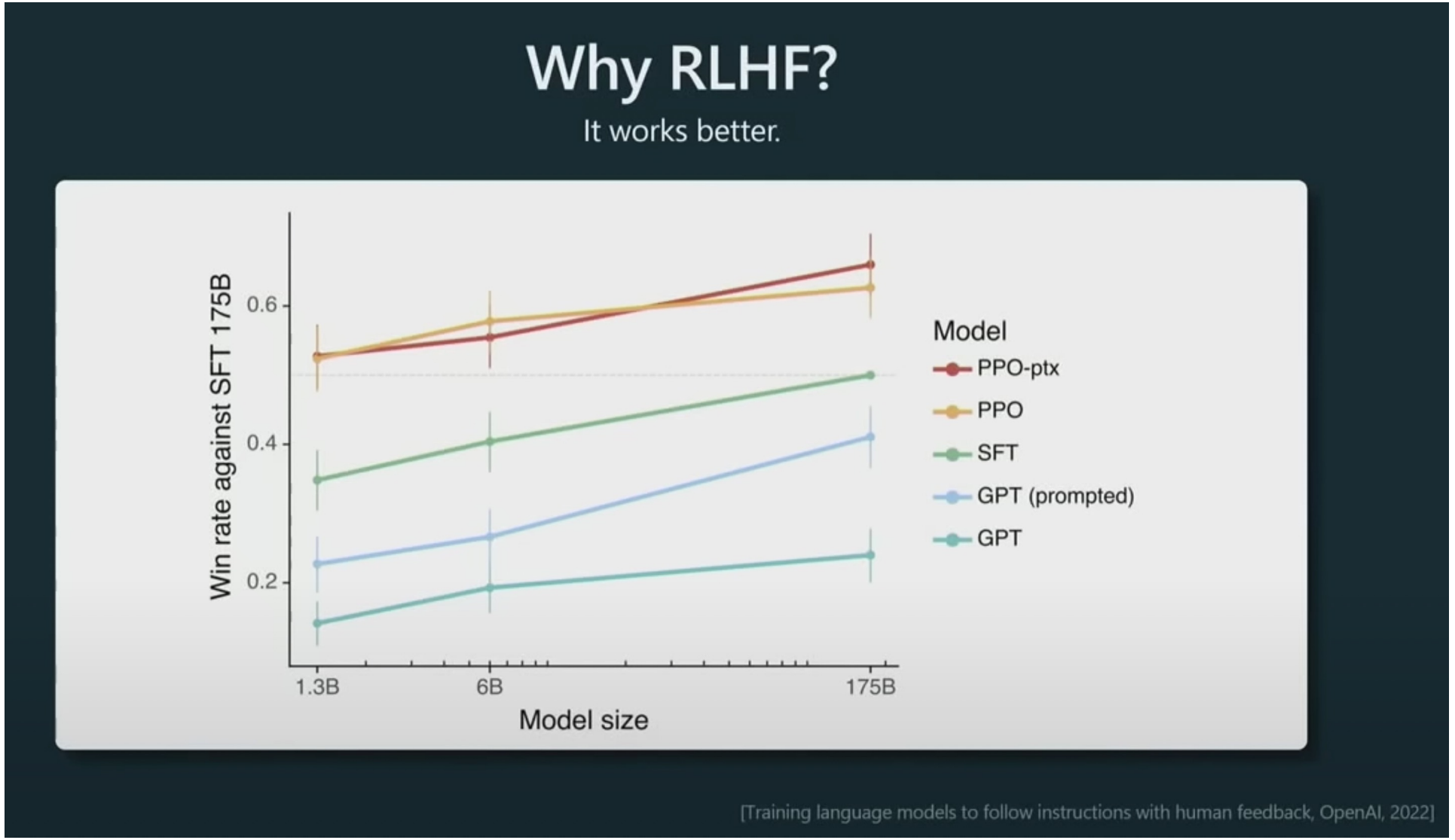


Reinforcement learning

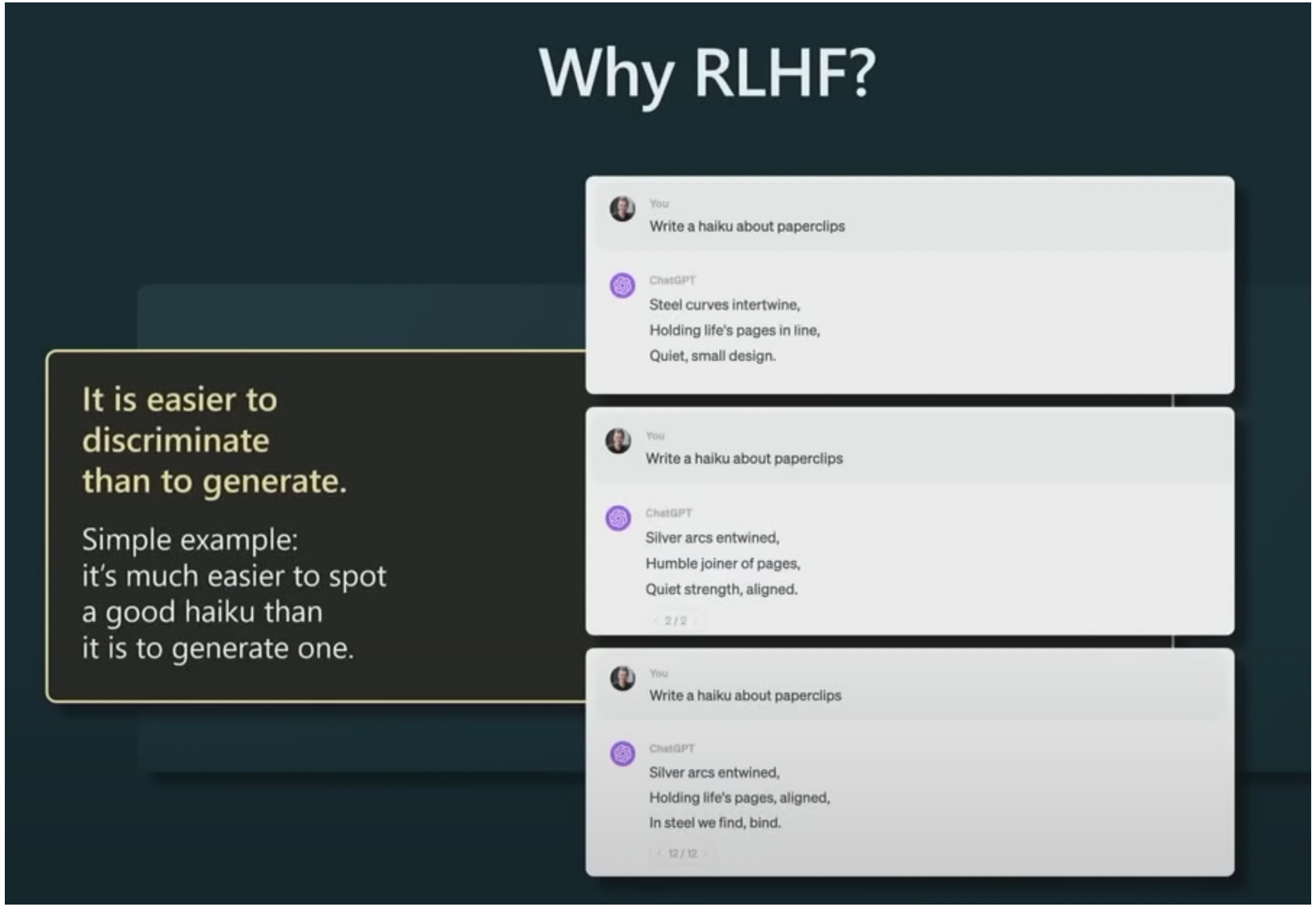
RL makes the model learn to generate responses with great scores



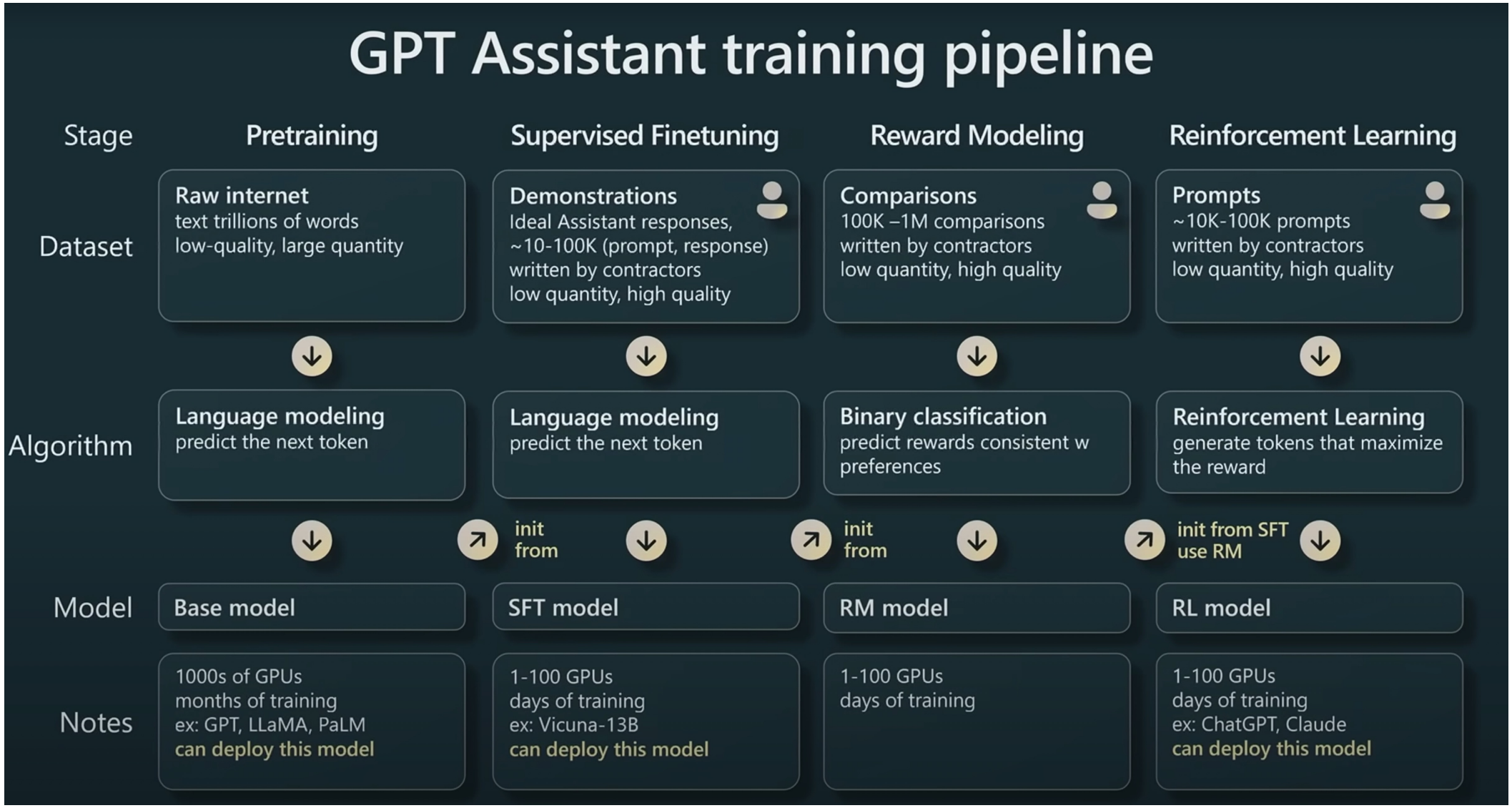
Reinforcement learning



Reinforcement learning



ChatGPT training pipeline



Source: Andrej Karpathy, State of GPT

Assistant models in the wild

北京大学 PEKING UNIVERSITY

Rank	Model	Elo Rating	Description	License
1	🥇 GPT-4	1274	ChatGPT-4 by OpenAI	Proprietary
2	🥈 Claude-v1	1224	Claude by Anthropic	Proprietary
3	🥉 GPT-3.5-turbo	1155	ChatGPT-3.5 by OpenAI	Proprietary
4	Vicuna-13B	1083	a chat assistant fine-tuned from LLaMA on user-shared conversations by LMSYS	Weights available; Non-commercial
5	Koala-13B	1022	a dialogue model for academic research by BAIR	Weights available; Non-commercial
6	RWKV-4-Raven-14B	989	an RNN with transformer-level LLM performance	Apache 2.0
7	Oasst-Pythia-12B	928	an Open Assistant for everyone by LAION	Apache 2.0
8	ChatGLM-6B	918	an open bilingual dialogue language model by Tsinghua University	Weights available; Non-commercial
9	StableLM-Tuned-Alpha-7B	906	Stability AI language models	CC-BY-NC-SA-4.0
10	Alpaca-13B	904	a model fine-tuned from LLaMA on instruction-following demonstrations by Stanford	Weights available; Non-commercial
11	FastChat-T5-3B	902	a chat assistant fine-tuned from FLAN-T5 by LMSYS	Apache 2.0
12	Dolly-V2-12B	863	an instruction-tuned open large language model by Databricks	MIT
13	LLaMA-13B	826	open and efficient foundation language models by Meta	Weights available; Non-commercial

A short summary

北京大学 PEKING UNIVERSITY

- We discuss the pipeline to train ChatGPT
- SFT, RM, and RL are critical to transform GPT to ChatGPT
- SFT, RM, and RL are also critical to transform GPT to **your own personalized assistant**

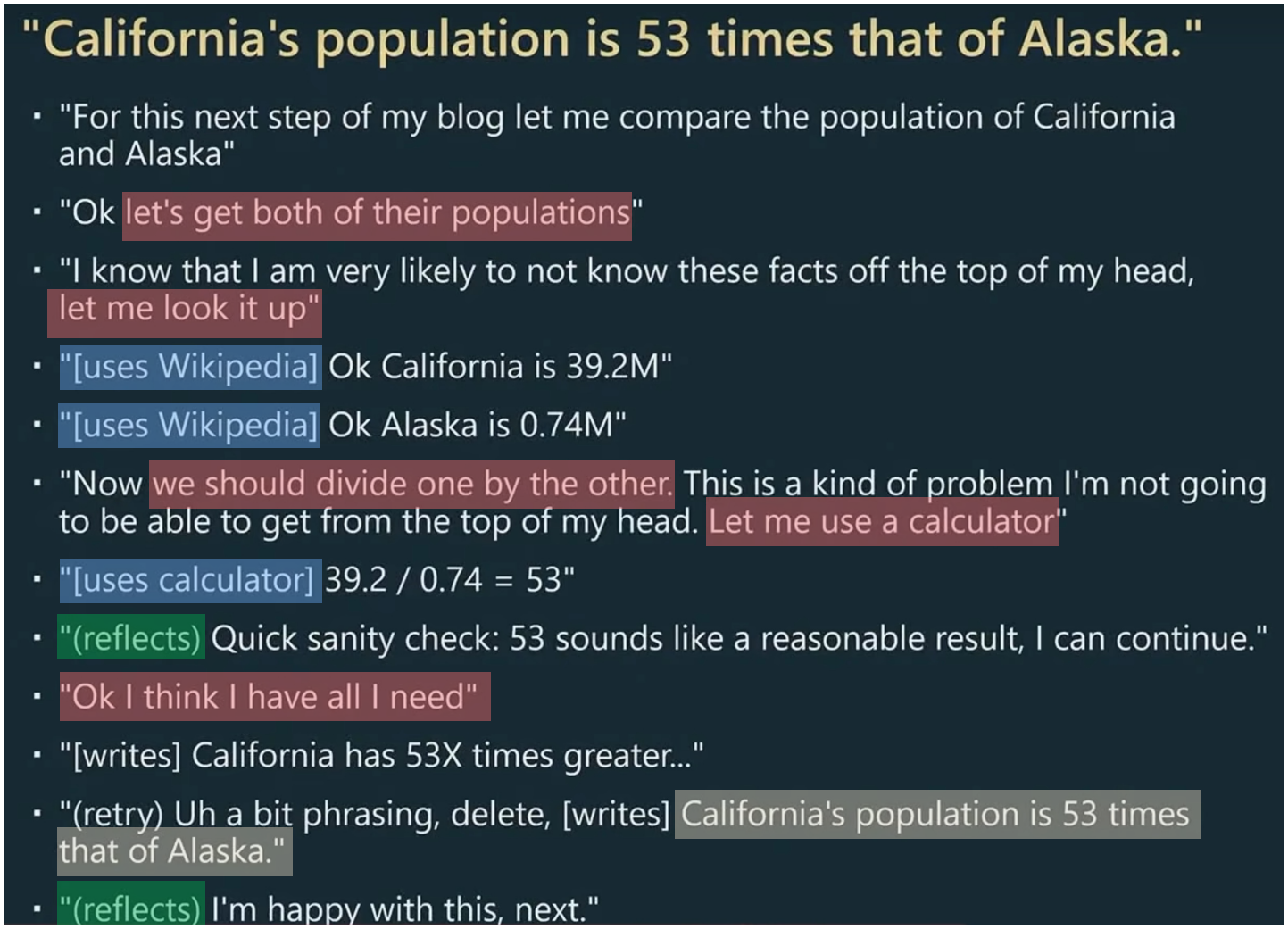
PART 03

Use LLM Effectively As Your Personal Copilot

Understand how human and LLM work differently

北京大学 PEKING UNIVERSITY

- Human can plan and reflect
- Human can use tools
- Human typically thinks more



Understand how human and LLM work differently

- LLM strips away all human behavior

- All of the internal monologue is stripped away in the text LLMs train on
- They spend the ~same amount of compute on every token
- => LLMs don't reproduce this behavior by default!
- They don't know what they don't know, they imitate the next token
- They don't know what they are good at or not, they imitate the next token
- They don't reflect. They don't sanity check. They don't correct their mistakes along the way
- They don't have a separate "inner monologue stream in their head"
- They do have very large fact-based knowledge across a vast number of areas
- They do have a large and ~perfect "working memory" (their context window)

Use prompt to help LLM work like a human

- **Chain of thoughts**: break up tasks into multiple steps/stages

(i) Zero-Shot

Q: There were 10 friends playing a video game online when 7 players quit. If each player left had 8 lives, how many lives did they have total?

A: The answer is

(Output) 80. ✗

(ii) Zero-Shot-CoT

Q: There were 10 friends playing a video game online when 7 players quit. If each player left had 8 lives, how many lives did they have total?

A: Let's think step by step.

(Output) There were 10 friends playing a video game online. This means that, at the start, there were 10 x 8 = 80 lives in total. Then, 7 players quit. This means that 7 x 8 = 56 lives were lost. Therefore, the total number of lives remaining is 80 - 56 = 24. The answer is 24. ✓

(iii) Manual-CoT

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

... ...

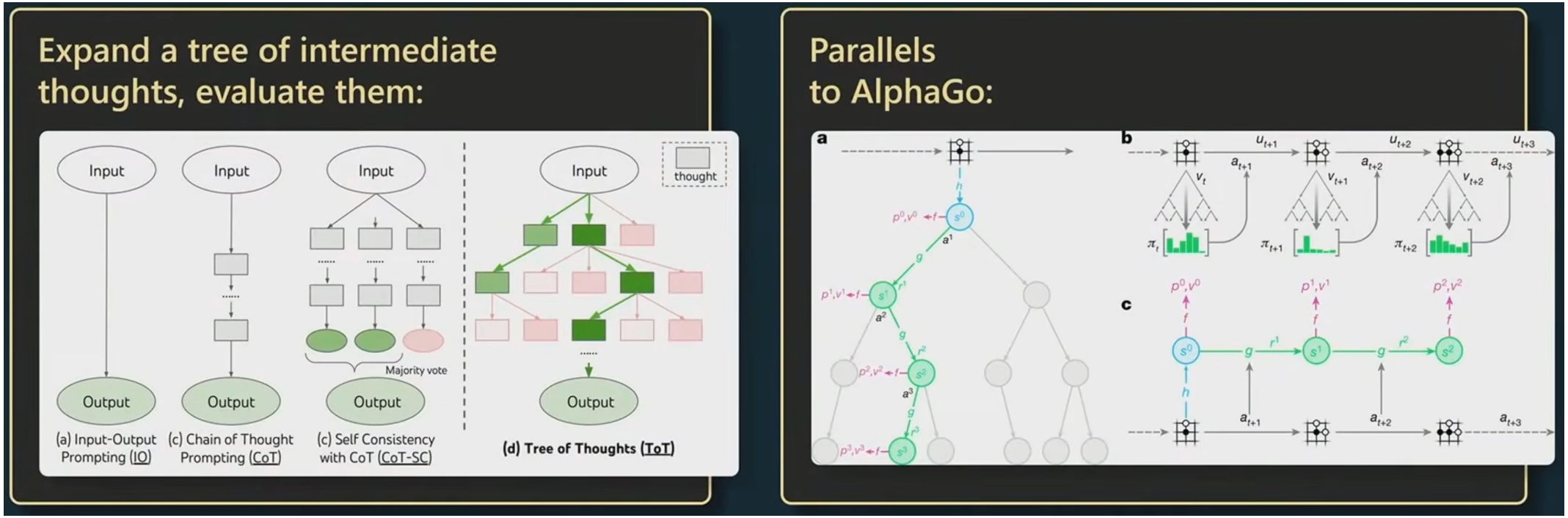
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 x 3 = 15 dollars. So she has 23 - 15 dollars left. 23 - 15 is 8. The answer is 8.

Q: In a video game, each enemy defeated gives you 7 points. If a level has 11 enemies total and you destroy all but 8 of them, how many points would you earn?
A:

(Output) You would earn 7 points for each enemy you defeat. So you would earn 7 * 11 = 77 points.

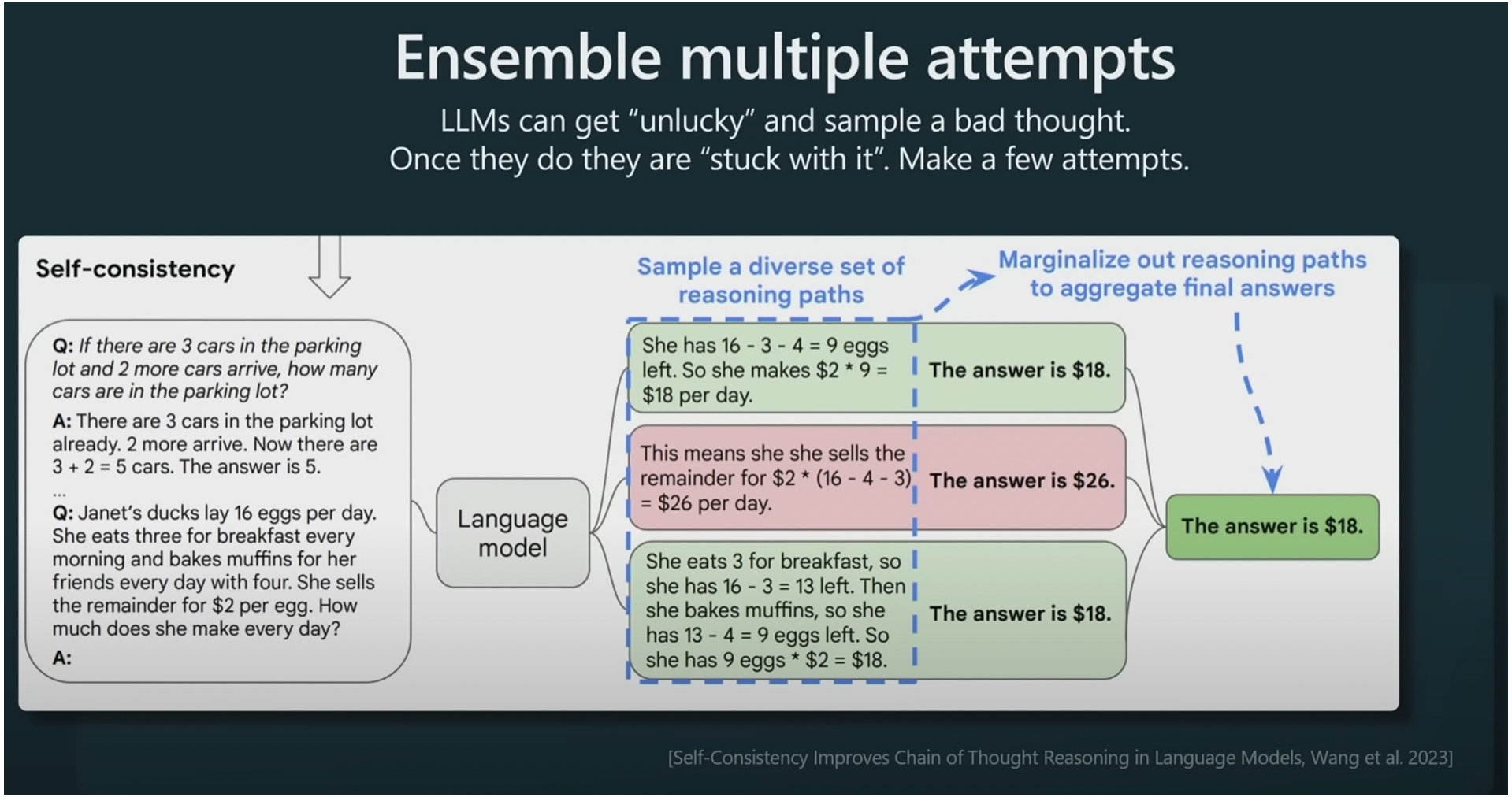
Tree of thought

- **Tree of thoughts**: expand thoughts, evaluate them and then go deeper

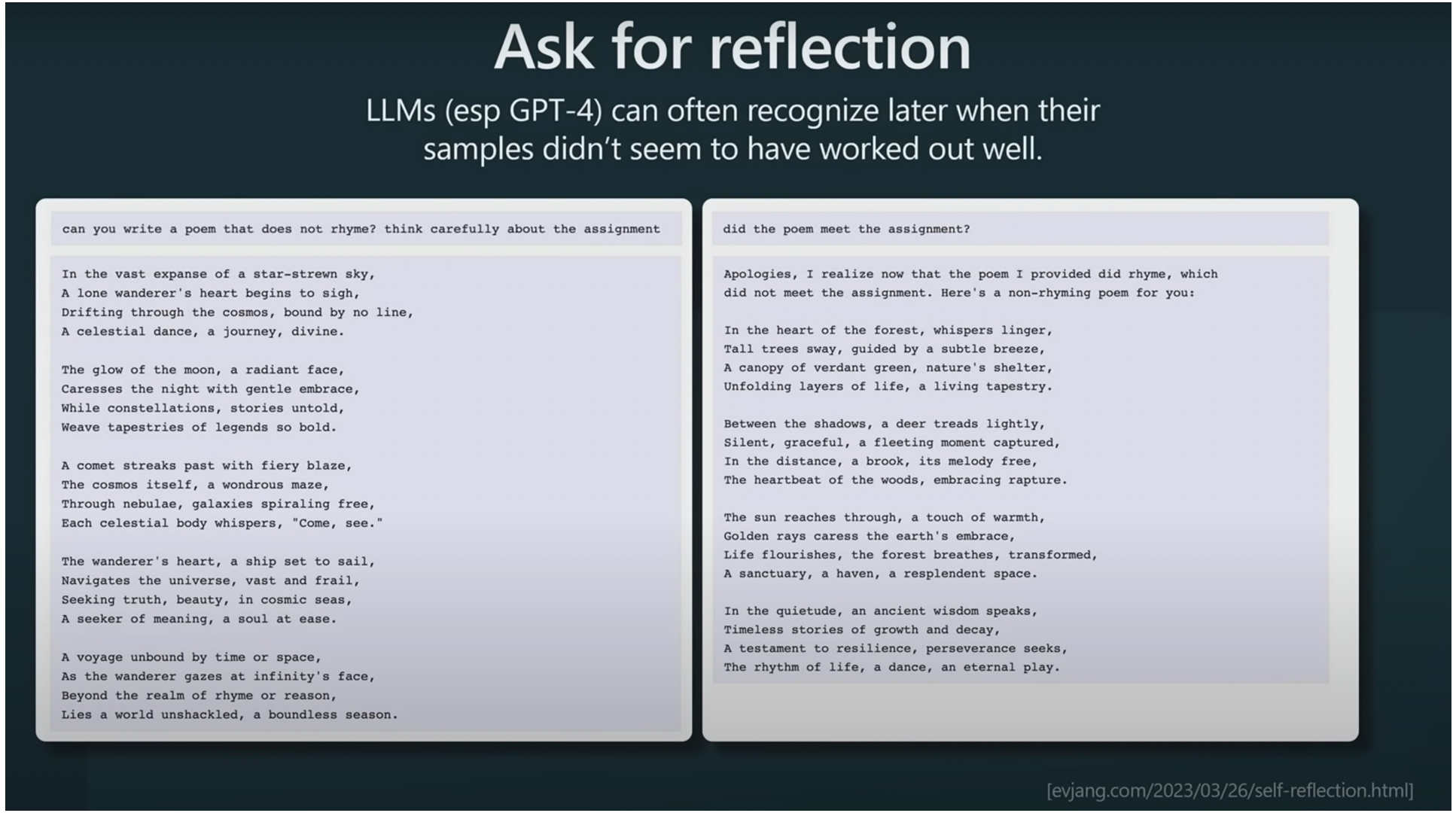


- How to find simple and effective prompts is still a hot research topic

Prompt ensemble



Ask for reflection



Automatic prompt engineering (APE)

北京大学 PEKING UNIVERSITY

- Learn a good prompt automatically

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	APE	Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

Related examples:

"You are a leading expert on this topic"

"Pretend you have IQ 120"

...

[Large language models are human-level prompt engineers, 2023]

RAG empowered LLM

LLM uses Bing Search

Retrieval-augmented generation (RAG) helps LLM generate more precise, up-to-date, and personalized contents.

你

请介绍一下北大

Copilot

北京大学，简称**北大**，创立于1898年，是中国近现代第一所国立综合性大学[1][2][3]。它的前身是**京师大学堂**，初期是中国最高学府，同时也是国家最高教育行政机关，负责全国教育的管理。辛亥革命后，校名改为**北京大学校**，后又改名为**国立北京大学**[2]。

北京大学位于北京市海淀区颐和园路5号，是一所享有盛誉的国立大学。它以其卓越的学术研究、丰富的历史文化和杰出的校友而闻名于世。[1][2][3]

!北京大学

了解详细信息 1 pku.edu.cn 2 zh.wikipedia.org 3 pku.edu.cn 4 gotopku.cn 5 bing.com 6 ccj.pku.edu.cn 7 ja.wikipedia.org

RAG empowered LLM

北京大学 PEKING UNIVERSITY

Copilot

Chinchilla Law，或称为Chinchilla Scaling Laws，是与神经网络相关的一种缩放定律。让我为您详细解释一下：

1. **神经模型的参数大小**：通常，神经模型的大小可以通过其参数数量来衡量。然而，在使用稀疏模型（例如专家混合模型）时，情况会稍有复杂。在稀疏模型中，每次推断时只使用部分参数。相比之下，其他类型的神经网络（如Transformer网络）在每次推断时始终使用所有参数。
2. **训练数据集的大小**：训练数据集的大小通常通过其中包含的数据点数量来量化。较大的训练数据集通常更受青睐，因为它们为模型提供了更丰富、更多样的信息，有助于提高模型对未见数据的泛化性能。然而，增加训练数据集的大小也会增加模型训练所需的计算资源和时间。
3. **训练成本**：训练成本通常以时间（训练模型所需的时间）和计算资源（训练模型所需的处理能力和内存）来衡量。值得注意的是，通过高效的训练算法、优化的软件库以及在GPU或TPU等专用硬件上的并行计算，可以显著降低训练成本。
4. **性能**：神经模型的性能是根据其在给定输入数据下准确预测输出的能力来评估的。常用于评估模型性能的指标包括准确率、F1分数、AUC-ROC等。

总之，Chinchilla Scaling Laws帮助我们理解神经网络中不同参数之间的关系，从而优化模型的训练和性能。 1 2 3 4

!Chinchilla

了解详细信息 1 en.wikipedia.org 2 jonkrohn.com 3 zhuanlan.zhihu.com 4 arxiv.org 5 planetchinchilla.com

RAG Bing Copilot

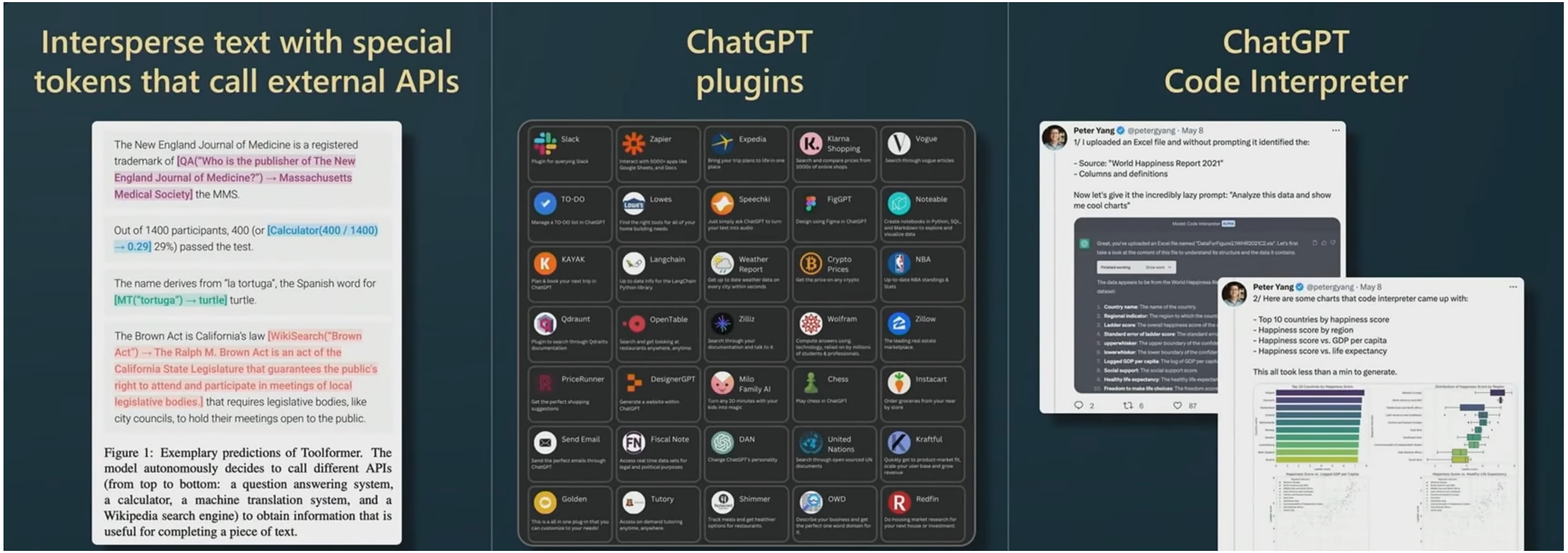
ChatGPT

根据我的知识，没有一个广泛认可的概念或术语称为“Chinchilla Law”。Chinchilla可能是指毛丝鼠，但“Chinchilla Law”在法律体系或法学中并不是一个常见的术语。如果您有关于“Chinchilla Law”可能涉及的特定背景或领域的进一步信息，请提供，我将尽力为您提供更多帮助。

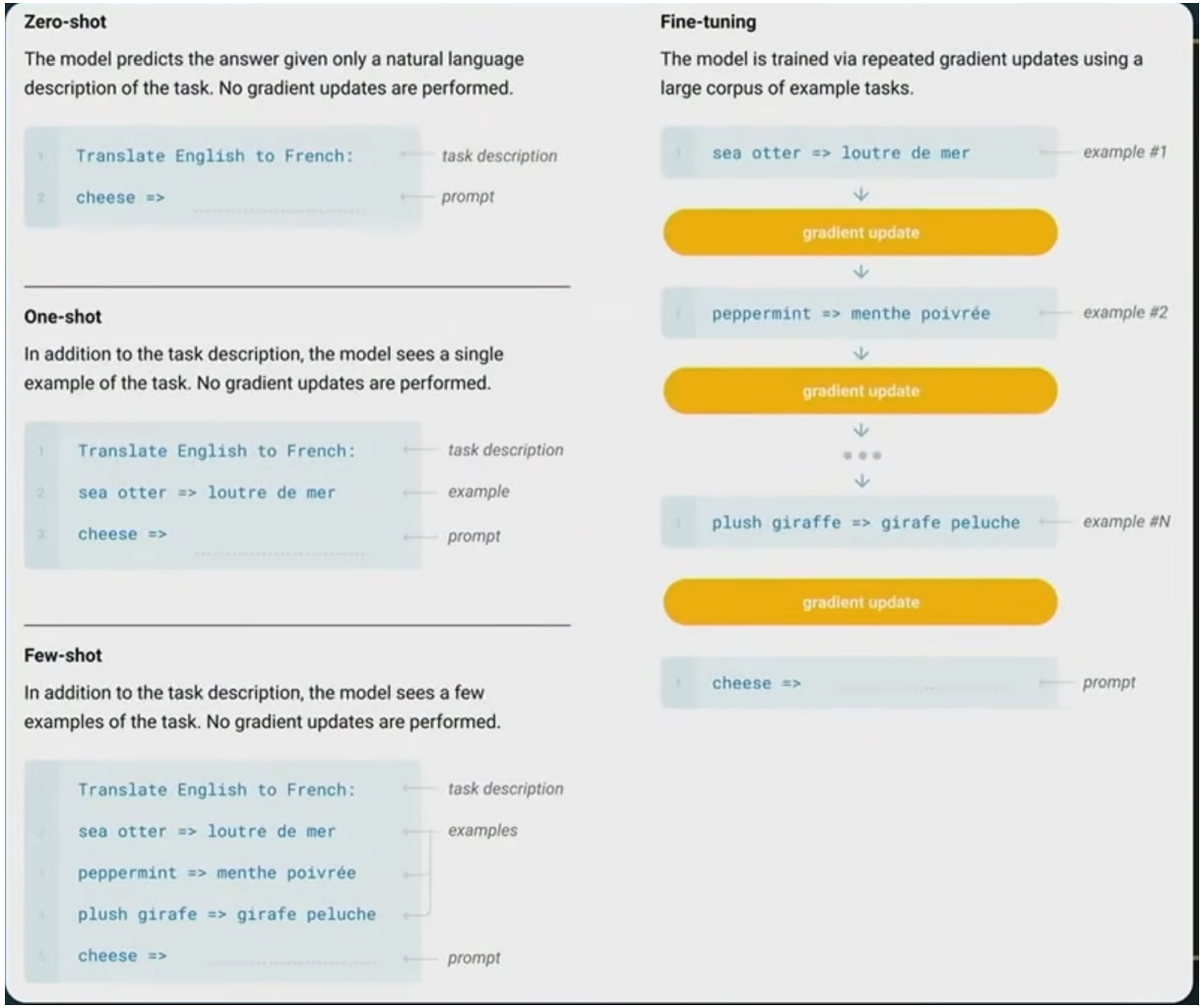
ChatGPT 3.5

Tool use

Offload tasks that LLM are not good at.



Finetuning



SFT and RLHF are all finetuning the base pretrained model

It is becoming a lot more accessible to finetune LLMs:

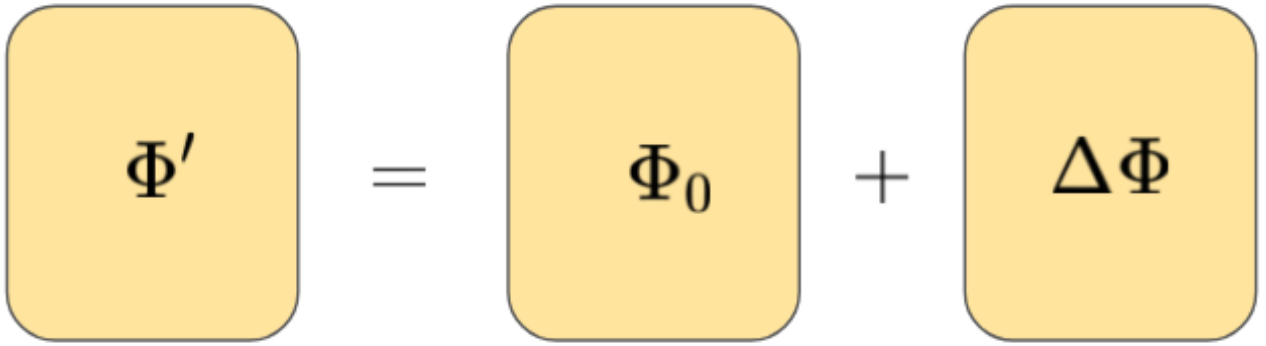
- Parameter Efficient FineTuning (PEFT), e.g. LoRA
- Low-precision inference, e.g. bitsandbytes
- Open-sourced high quality base models, e.g. LLaMA

Keep in mind:

- Requires a lot more technical expertise
- Requires contractors and/or synthetic data pipelines
- A lot slower iteration cycle
- SFT is achievable
- RLHF is research territory

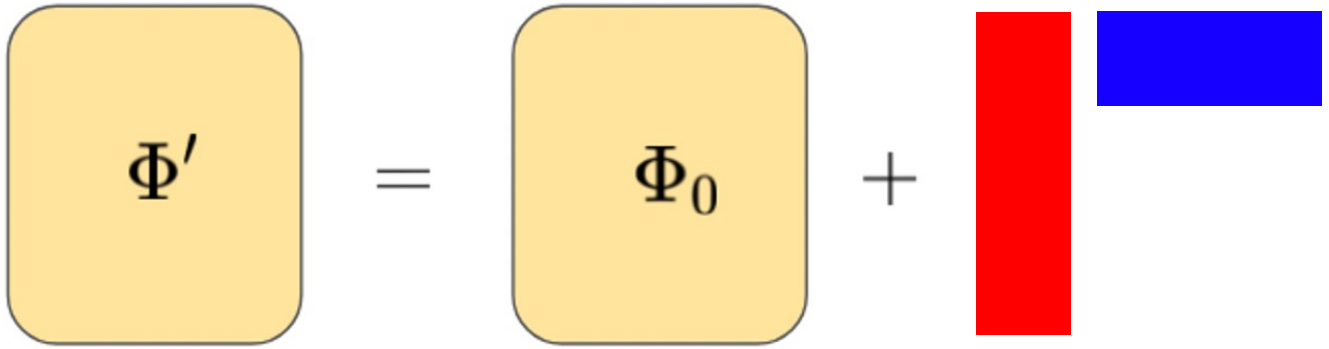
LoRA: Low-rank adaptation

Finetune: Inject weights to base model



Fine-tuned weight | Base model weight | Additional weight

LoRA: low-rank adaptation



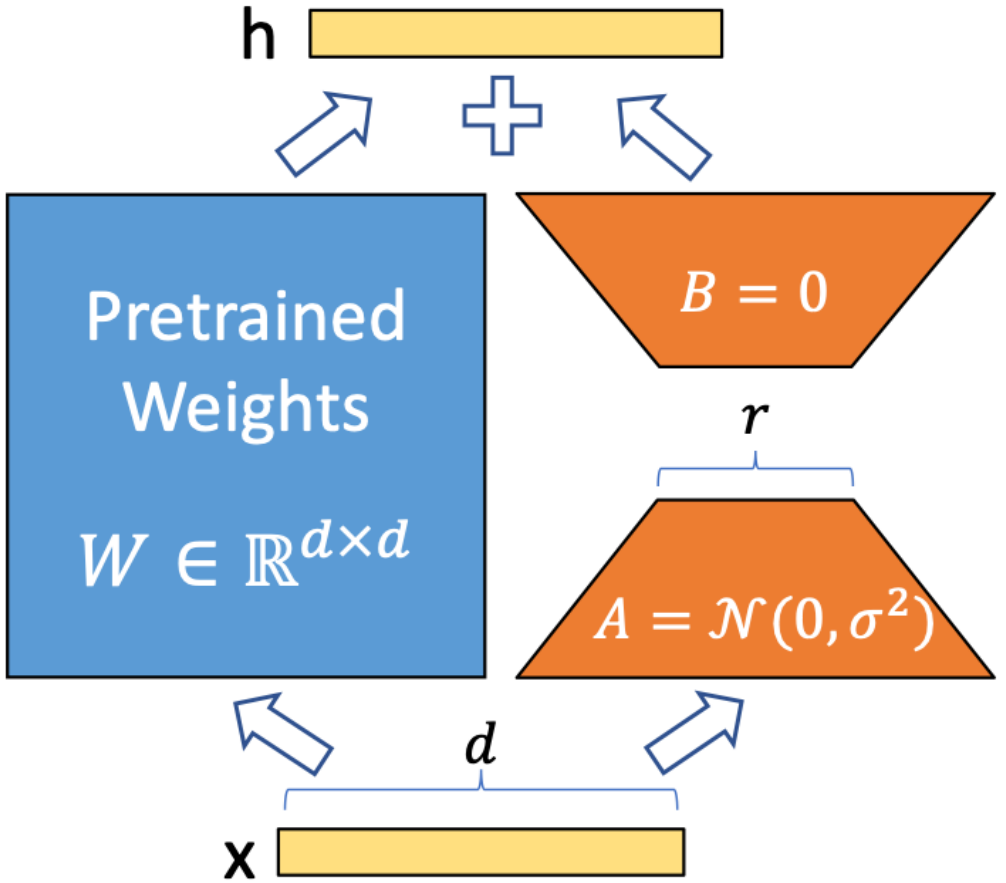
Fine-tuned weight | Base model weight | Low-rank weight

LoRA: Low-rank adaptation

$$W \Leftarrow W_0 + \Delta W$$

$$W_0 + \Delta W = W_0 + BA$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$



LoRA: Low-rank adaptation

北京大学 PEKING UNIVERSITY

Light but powerful

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

LoRA: Low-rank adaptation

北京大学 PEKING UNIVERSITY

Reference

E. J. Hu et. al., LoRA: Low-Rank Adaptation of Large Language Models, https://arxiv.org/abs/2106.09685

Q. Zhang et. al., LoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning, https://arxiv.org/abs/2303.10512

How to use LLM effectively?

北京大学 PEKING UNIVERSITY

Recommendations from OpenAI

Goal 1: Achieve your top possible performance

- Use GPT-4
- Use prompts with detailed task context, relevant information, instructions
  - *"what would you tell a task contactor if they can't email you back?"*
- Retrieve and add any relevant context or information to the prompt
- Experiment with prompt engineering techniques (previous slides)
- Experiment with few-shot examples that are 1) relevant to the test case, 2) diverse (if appropriate)
- Experiment with tools/plugins to offload tasks difficult for LLMs (calculator, code execution, ...)
- Spend quality time optimizing a pipeline / "chain"
- If you feel confident that you maxed out prompting, consider SFT data collection + finetuning
- Expert / fragile / research zone: consider RM data collection, RLHF finetuning

Goal 2: Optimize costs

- Once you have the top possible performance, attempt cost saving measures (e.g. use GPT-3.5, find shorter prompts, etc.)

Use cases

Models may be biased

Models may fabricate ("hallucinate") information

Models may have reasoning errors

Models may struggle in classes of applications, e.g. spelling related tasks

Models have knowledge cutoffs (e.g. September 2021)

Models are susceptible to prompt injection, "jailbreak" attacks, data poisoning attacks,...

Recommendations:

→ Use in low-stakes applications, combine with human oversight

→ Source of inspiration, suggestions

→ Copilots over autonomous agents

Some of these issues have been addressed nowadays

北京大学
PEKING UNIVERSITY

Thank you!

Kun Yuan homepage: https://kunyuan827.github.io/