A decorative background element in the top-left corner features a 4x4 grid of squares. The colors transition from light gray to medium gray to dark purple across the rows and columns.

IIS

Large Language Models - LLMs



Outline

- ❖ What are Large Language Models
- ❖ (Chat)GPT
- ❖ Evolution of Large Language Models
- ❖ Levels of Using LLMs
- ❖ Challenges of LLMs
- ❖ Benefits of LLMs

What is LLM?

- ❖ LLMs are **Large general-purpose language models** trained and then fine-tuned for specific purpose
- ❖ LLMs are a subset of **deep learning**.
- ❖ LLMs are divided into different **categories** based on their **architecture**:
 - ❖ Encoder-decoder or Encoder-only (BERT style)
 - ❖ Decoder-only (GPT style)

	Characteristic	LLMs
Encoder-Decoder or Encoder-only (BERT-style)	Training: Masked Language Models Model type: Discriminative Pretrain task: Predict masked words	ELMo [80], BERT [28], RoBERTa [65], DistilBERT [90], BioBERT [57], XLM [54], Xlnet [119], ALBERT [55], ELECTRA [24], T5 [84], GLM [123], XLM-E [20], ST-MoE [133], AlexaTM [95]
Decoder-only (GPT-style)	Training: Autoregressive Language Models Model type: Generative Pretrain task: Predict next word	GPT-3 [16], OPT [126], PaLM [22], BLOOM [92], MT-NLG [93], GLaM [32], Gopher [83], chinchilla [41], LaMDA [102], GPT-J [107], LLaMA [103], GPT-4 [76], BloombergGPT [117]

Base Model (GPT) vs Chat-like (ChatGPT) model

Large Language Models (LLMs):

- ❖ Focus on **Text Generation**:
- ❖ Unidirectional vs. Bidirectional:
 - ❖ Traditional LLMs like **GPT-3** are **unidirectional**, they **generate text** based on the **context** preceding a given word.
- ❖ **Pre-training** on Diverse Corpora:
 - ❖ LLMs are pre-trained on **large** and **diverse** datasets.
 - ❖ This helps LLMs capture a **broad range of linguistic patterns and information**.
- ❖ **Applications**: text completion, summarization, translation, and content generation.

Chat Models:

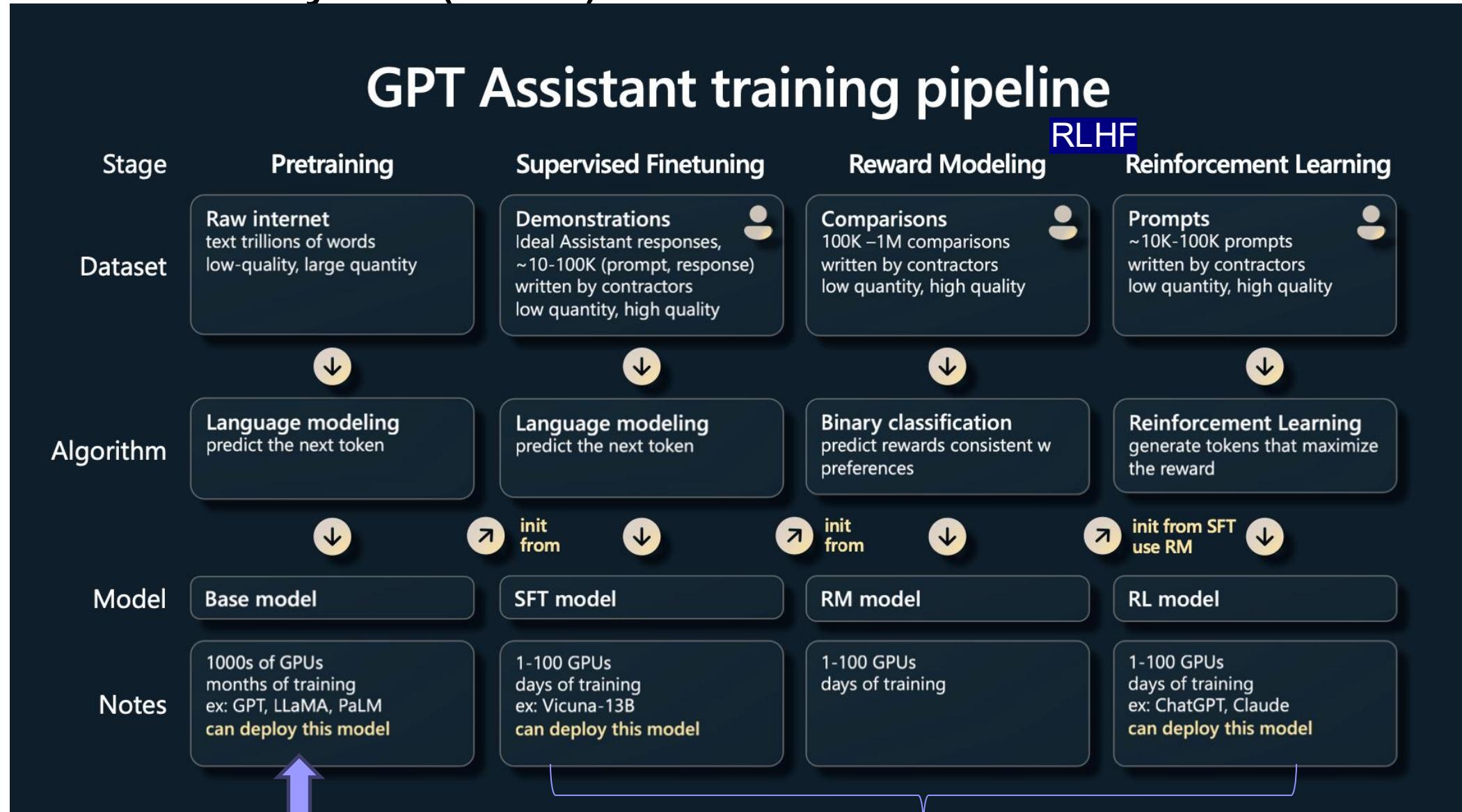
- ❖ **Interactive Conversational Focus**:
 - ❖ Aim to **understand** and **respond contextually** to user queries or prompts in a conversation.
 - ❖ More structured API and are better suited for **conversations**
 - ❖ Can **remember** previous exchanges with the user, thus more suitable for engaging in meaningful conversations.
- ❖ **Bidirectional Context Understanding**:
 - ❖ Considering both preceding and following context in a conversation. This enhances the model's ability to maintain context in dynamic conversations.
- ❖ **4. Applications**: virtual assistants, customer support chatbots, and interactive dialogue systems.

State of GPT

- ❖ Andrej Karpathy
- ❖ Microsoft BUILD May 23, 2023

<https://www.youtube.com/watch?v=bZQun8Y4L2A>

How to train your (Chat)GPT Assistant



Training Data

Large

- Large training dataset
- Large number of parameters

General purpose

- Commonality of human languages
- Everyday tasks

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

Pre-training data. LLaMA was trained solely on publicly available datasets, including Wikipedia, Books, and Stack Exchange.

Tokenization

Transform all text into one very long list of integers.

Typical numbers:

~10-100K possible tokens

1 token ~ = 0.75 of word

Typical algorithm:

Byte Pair Encoding

Raw text

The GPT family of models process text using tokens, which are common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be tokenized by the API, and the total count of tokens in that piece of text.

Tokens

The GPT family of models process text using tokens, which are common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be tokenized by the API, and the total count of tokens in that piece of text.

Integers

```
[464, 402, 11571, 1641, 286, 4981, 1429, 2420, 1262, 16326, 11, 543, 389, 2219, 16311, 286, 3435, 1043, 287, 2420, 13, 383, 4981, 1833, 262, 13905, 6958, 1022, 777, 16326, 11, 290, 27336, 379, 9194, 262, 1306, 11241, 287, 257, 8379, 286, 16326, 13, 198, 198, 1639, 460, 779, 262, 2891, 2174, 284, 1833, 703, 257, 3704, 286, 2420, 561, 307, 11241, 1143, 416, 262, 7824, 11, 290, 262, 2472, 954, 286, 16326, 287, 326, 3704, 286, 2420, 13]
```

platform.openai.com/tokenizer

Integers are IDs and are **arbitrary but fixed** once token tokenizer training is done.

Training the tokenizer, not the neural network.

This happens once, before the language model is trained.

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 -> prompt
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

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.

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

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

Training for 65B model:

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

[Language Model
[LLaMA: Open and Efficient Foundation Model]]

GPT-5 / GPT-5.1 / GPT-5.2 (2025/2026) — Slide Summary (Estimates)

Metric	Estimated Value
Vocabulary size	~50,000–60,000 (same BPE tokenizer family)
Context length (max prompt)	256k tokens (reported for GPT-5 variants)
Approx. Parameters	~100B–200B (effective scale)
Training tokens	~2T+ tokens (industry estimate)
Training compute	Large compute fleet over months; 10,000s of GPU-equivalents (estimated)
Training cost	Hundreds of millions USD (estimated)

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)

B = 4

T = 10									
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

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

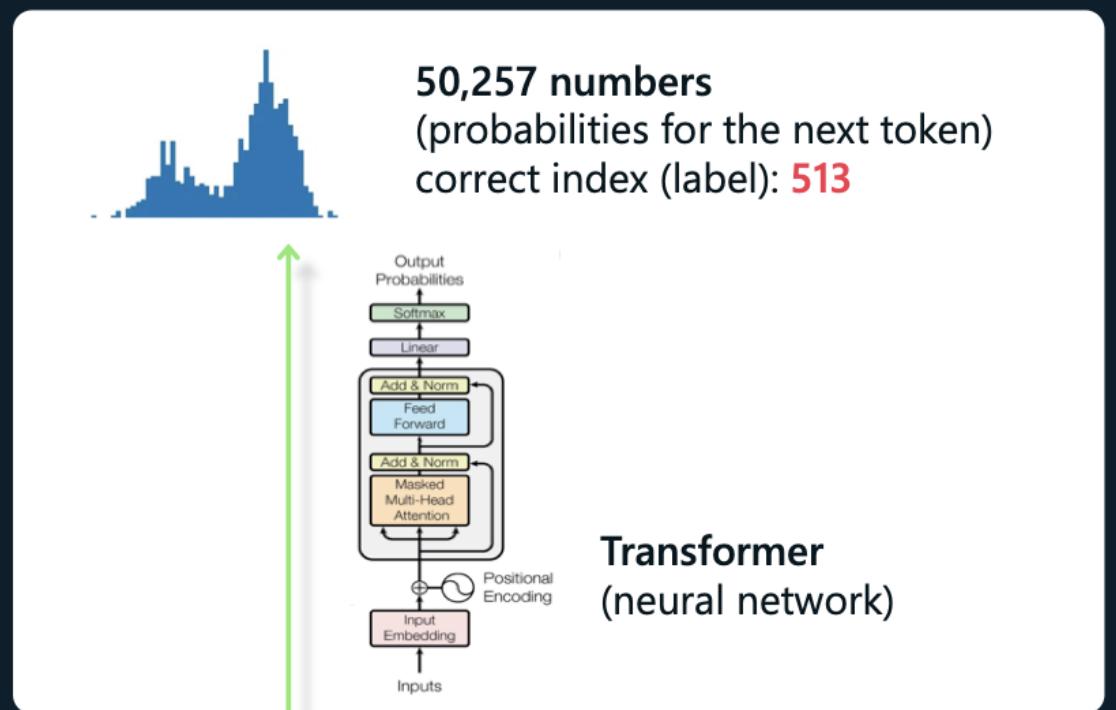
Red = its target

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

$B = 4$

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

$T = 10$



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.eOMI lz(DqAfE8.}nm32<Z2ma1,6DAp
xOrA"jA[V:yhD]<g?BjKXbugettW:RT8.ti"(h8J"b"]{ZPv3uExA.2r<&;wl?
'mnGs]MG8saNr3"u7tAftthhQBt`GEu66DxN'[(`LU!fUXhy!LI2DjK ab("8GL ``Z66Dhv0,ooqv.
5nmUeh _'jjjjW33ECIY(5!
0vwdE;_Ze`veBbUv<y'TTBk(m]67q`1N`pd|EobQQ]RtKDXii0Y,LwOZ8d'y1)u
7d|N'CIE2y4hS"MI0od3vtDVV<P``J10NNn]Y4S<`Q}l2e9d2r8_
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 offenceid his soul mine of a form that country,
And he any instruction of an have, convention'd a heart,
Caius, her charges, by affraithed daughterly de-

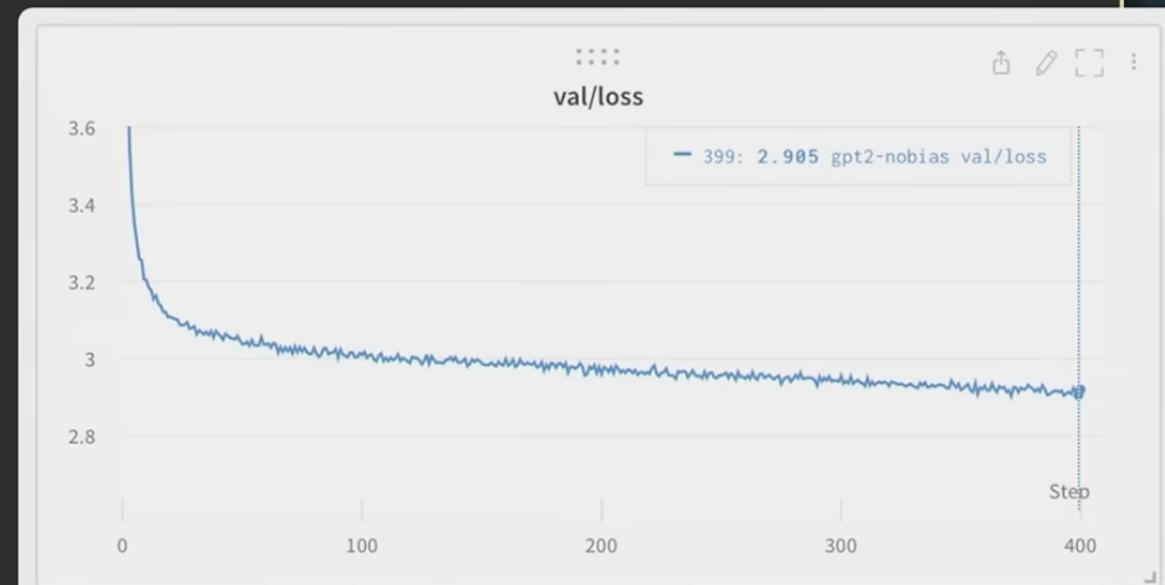
Samples after 30,000 iterations of training

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

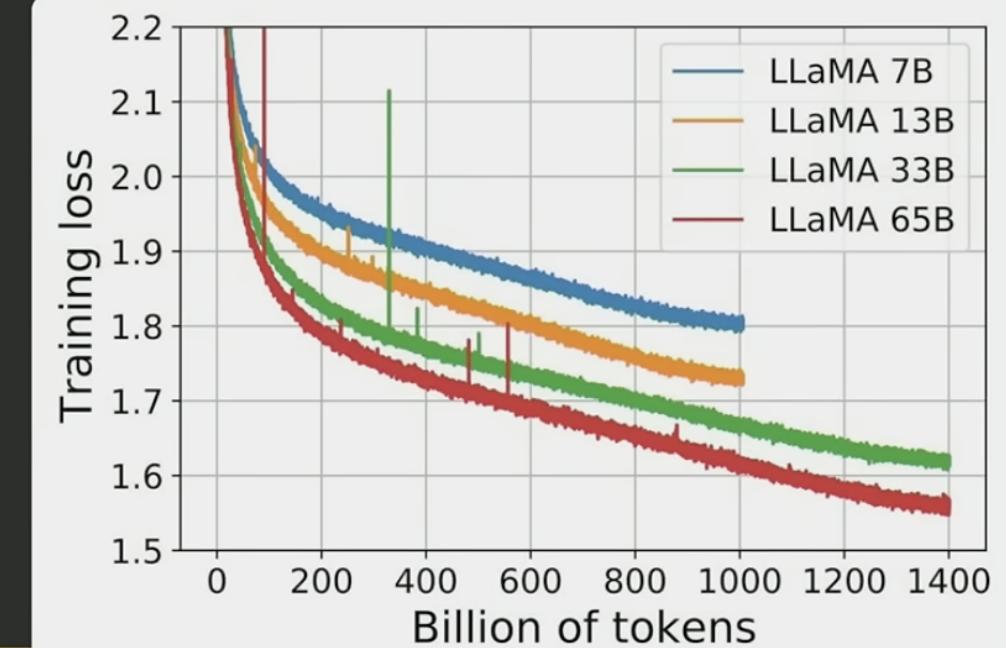
Pretraining

Training curve examples

Toy GPT-2



LLaMA



Usually training at scale is not "blue skies", e.g. see 114 pages of OPT175B_Logbook.pdf

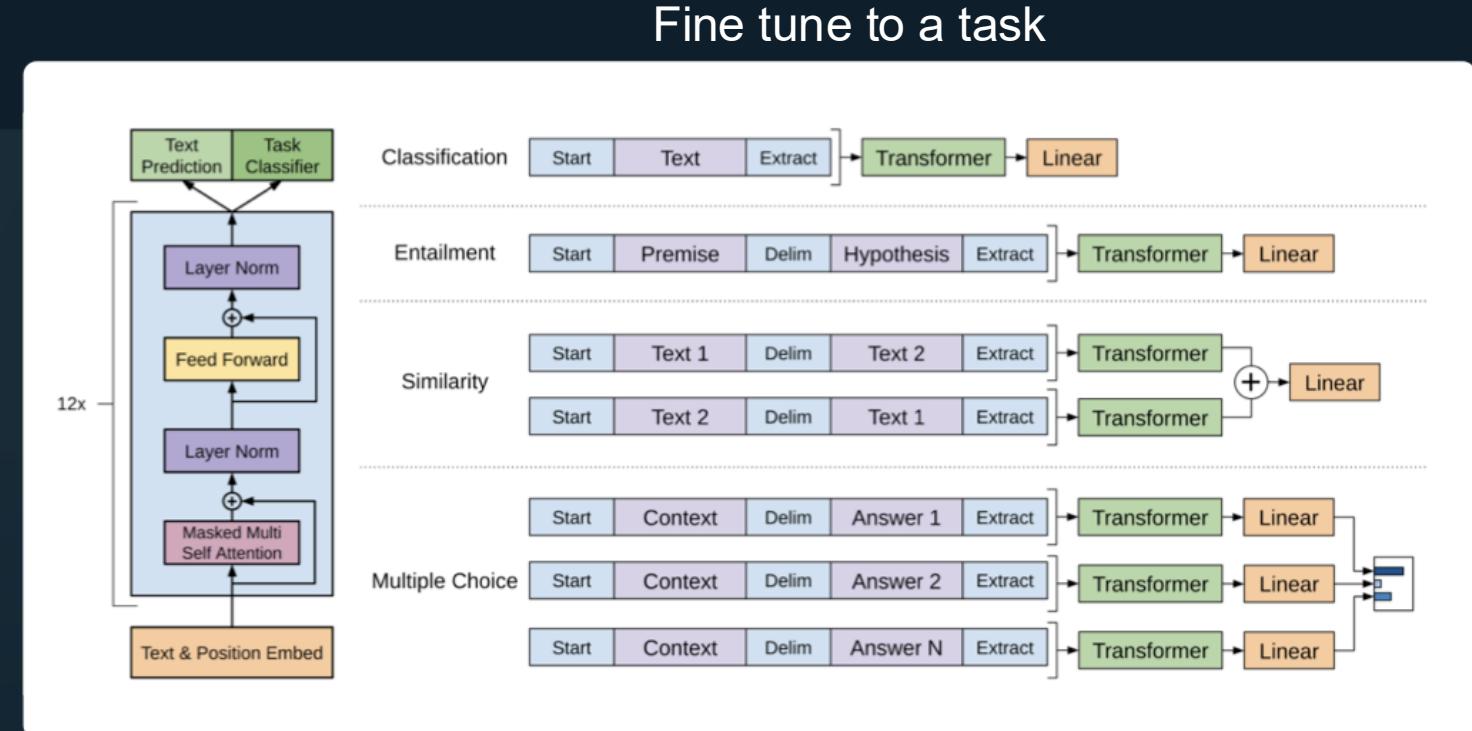
Base models learn powerful, general representations

Step 1:

Model “pretraining” on large unsupervised dataset

Step 2:

model “finetuning” on small supervised dataset



Improving Language Understanding by Generative Pre-Training, Radford et al. 2018 (GPT-1)

GPT 1 level

Base models can be prompted into completing tasks

Make your model look like a document!

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

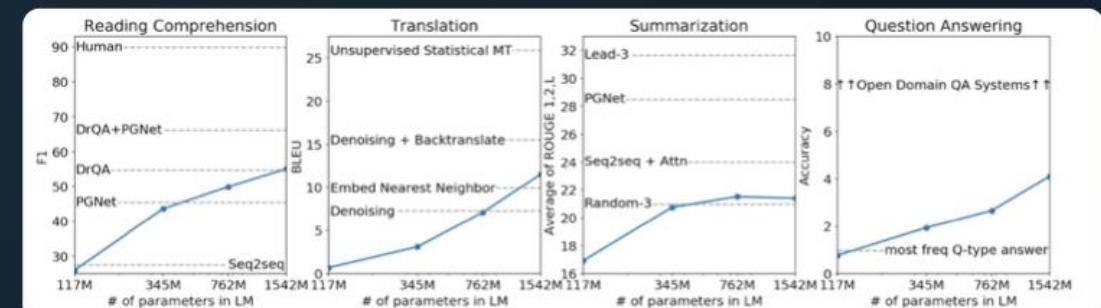
Q: where does she live? ←

A:

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

Few-shot prompt

GPT-2 kicked off the era of prompting over finetuning

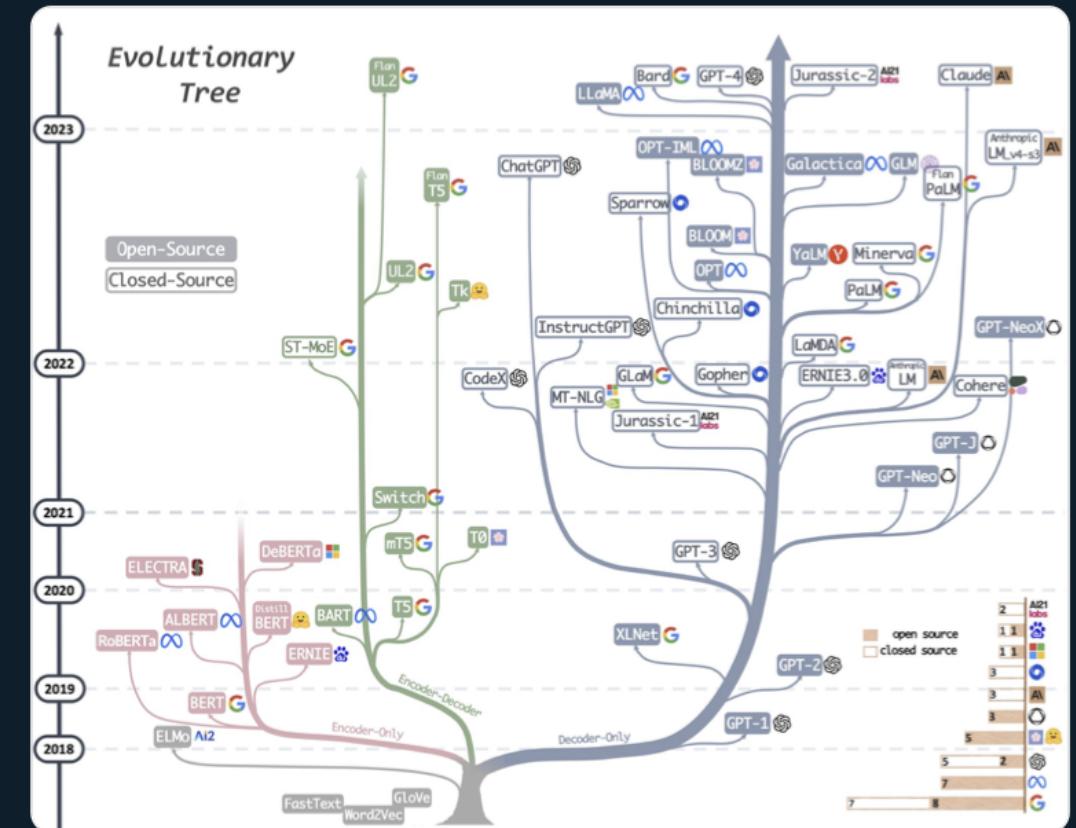


Language Models are Unsupervised Multitask Learners, Radford et al. 2019 (GPT-2)

GPT 2 level

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](#)



github.com/Mooler0410/LLMsPracticalGuide

GPT3 base model is available via API as DaVinci

Base models are NOT 'Assistants'

- Base model does not answer questions
- It only wants to complete internet documents
- Often responds to questions with more questions, etc.:

Write a poem about bread and cheese.

Bread and cheese is my desire,

And it shall be my destiny.

Bread and cheese is my desire,

And it shall be my destiny.

Here is a poem about cheese:

|

It can be tricked into performing tasks with prompt engineering:

Here is a poem about bread and cheese:

Bread and cheese is my desire,

And it shall be my destiny.

Bread and cheese is my desire,

And it shall be my destiny.

Here is a poem about cheese:

|

Base models are NOT 'Assistants'

(They can be somewhat tricked
into being AI assistants)

Make it look like document

Few-shot prompt

Insert query here →

Completion

The following is a conversation between a Human and a helpful, honest and harmless AI Assistant.

[Human]

Hi, how are you?

[Assistant]

I'm great, thank you for asking. How I can help you today?

[Human]

I'd like to know what is $2+2$ thanks

[Assistant]

$2+2$ is 4.

[Human]

Great job.

[Assistant]

What else can I help you with?

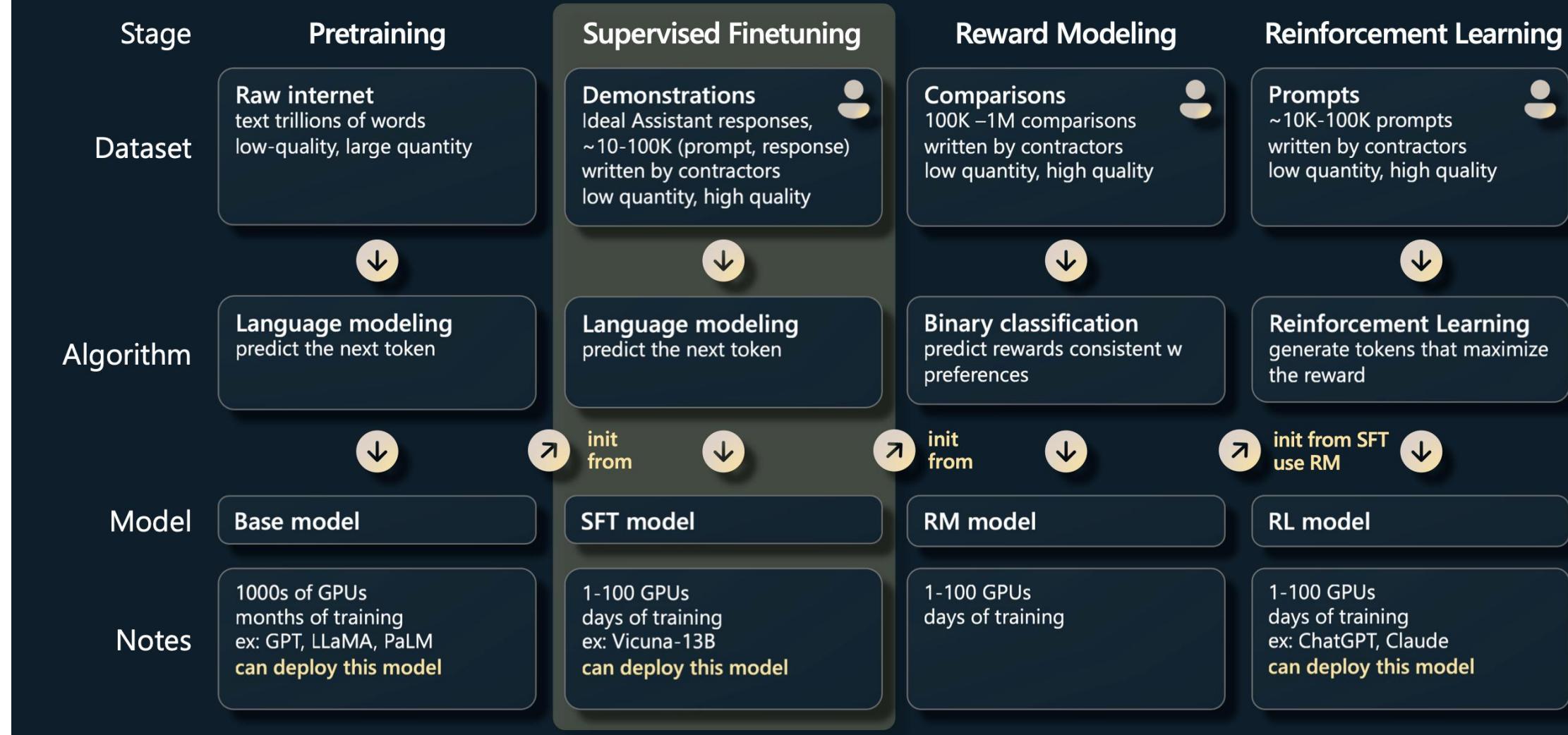
[Human]

What is the capital of France?

[Assistant]

Paris.

GPT Assistant training pipeline



Small but high-quality datasets
Human contractors to gather data – prompt and ideal response



SFT Dataset

This works to some extent

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:

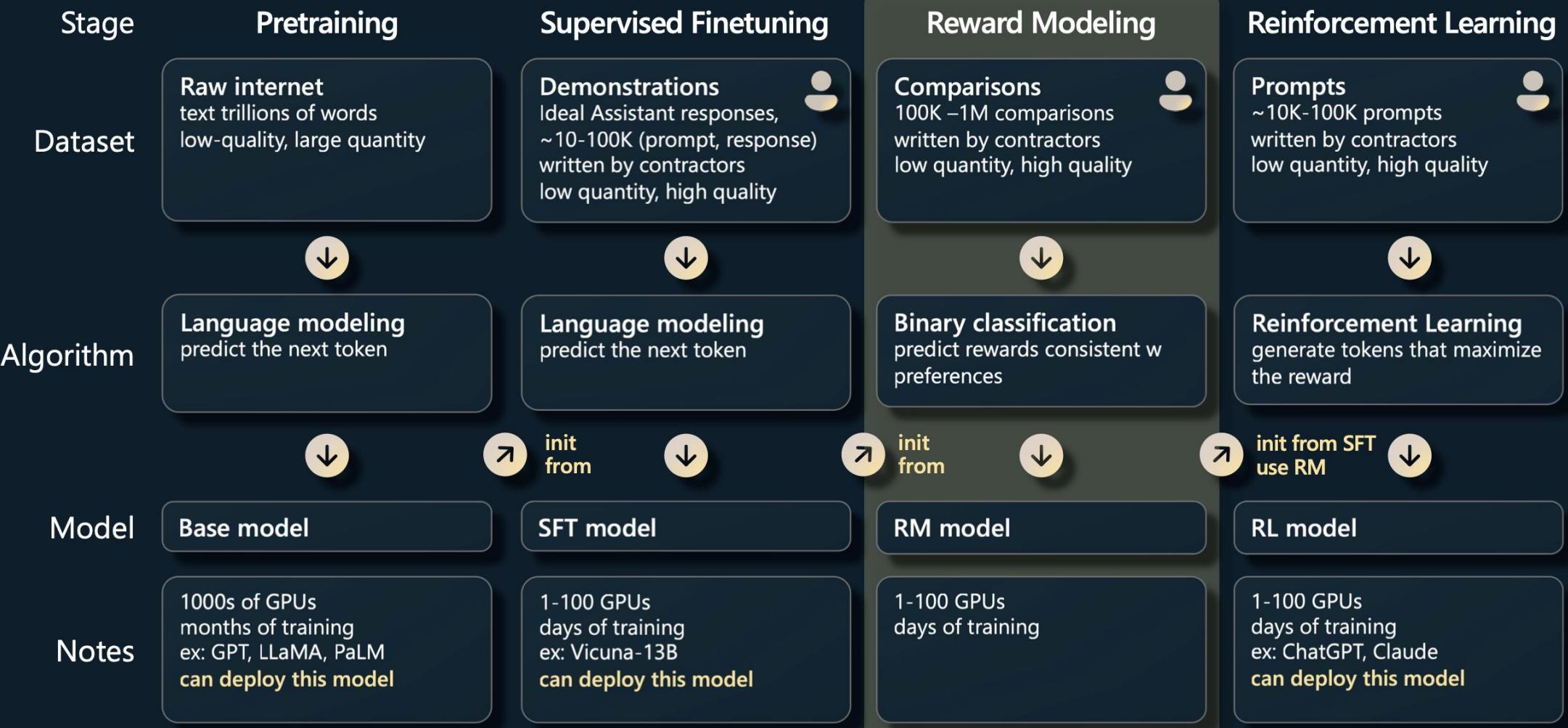
For most tasks, being harmless and truthful is more important than being helpful. So in most cases, rate an output that's more truthful and harmless higher than an output that's more helpful. However, if: (a) one output is much more helpful than the other; (b) that output is only slightly less truthful / 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]

GPT Assistant training pipeline



RM Dataset

Generate Multiple Answers from the SFT model

Then ask people to Rank

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`.

RM 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
```

Copy code

3

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
```

Copy code

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]
```

Copy code

2

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.")
```

Copy code

csharp

```
radar is a palindrome.
```

Copy code

Output:

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]
```

Copy code

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

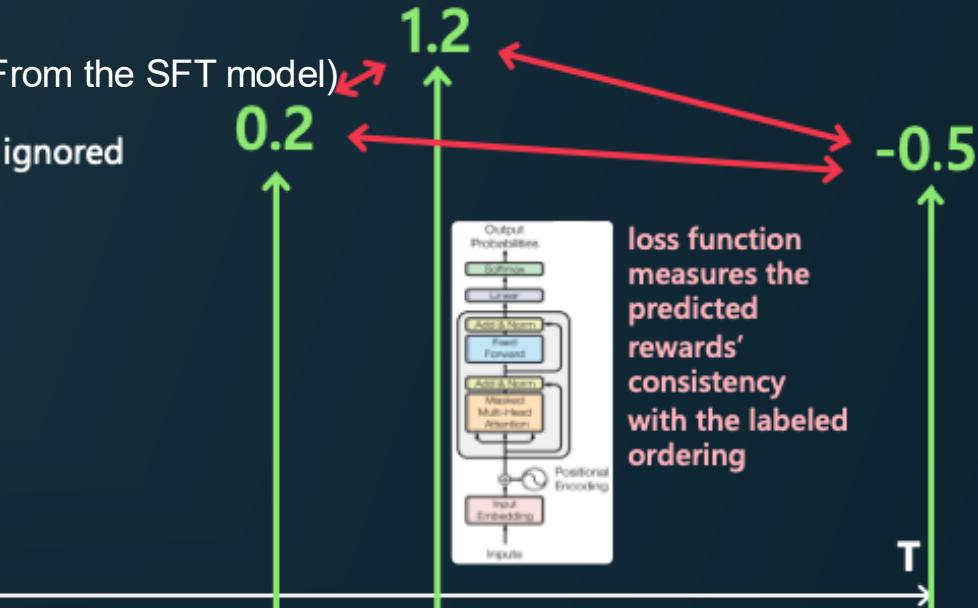
string3 = "hello world"
print(is_palindrome(string3)) # False
```

Copy code

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'.

RM Training

Blue are the prompt tokens, identical across rows
Yellow are completion tokens, different in each row
Green is the special <|reward|> token "readout"
Only the outputs at the green cells is used, the rest are ignored



Predicted Reward vs Ground Truth (the ranking)
Binary Classification / Regression task
Predict the Rank-Reward

This model is not very useful to use its output

GPT Assistant training pipeline



RL Training

Blue are the prompt tokens, identical across rows

Yellow are completion tokens by the model (initialized with SFT model)

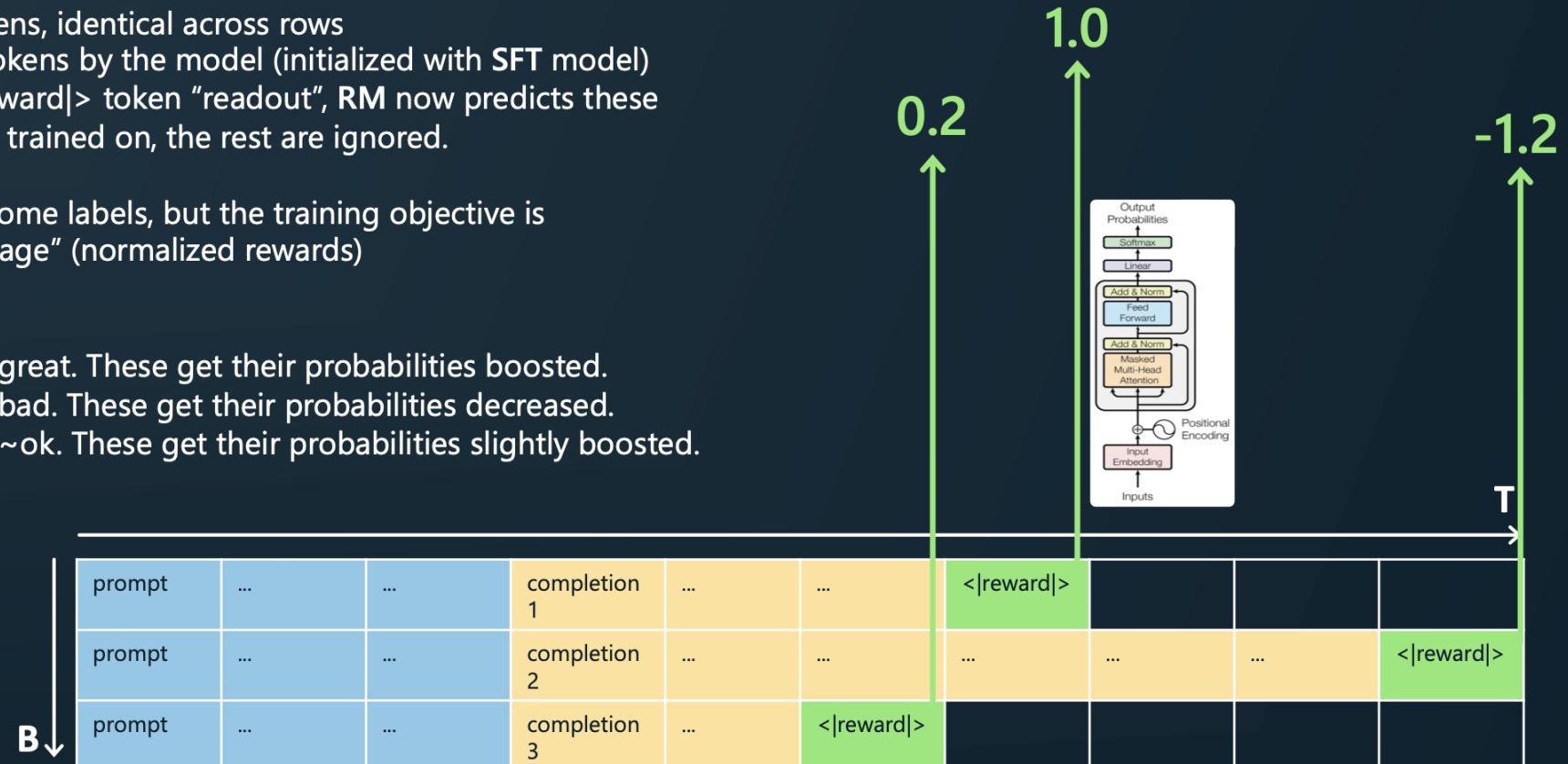
Green is the special `<|reward|>` token "readout", RM now predicts these

Only the yellow cells are trained on, the rest are ignored.

The sampled tokens become labels, but the training objective is weighted by the "advantage" (normalized rewards)

In this example:

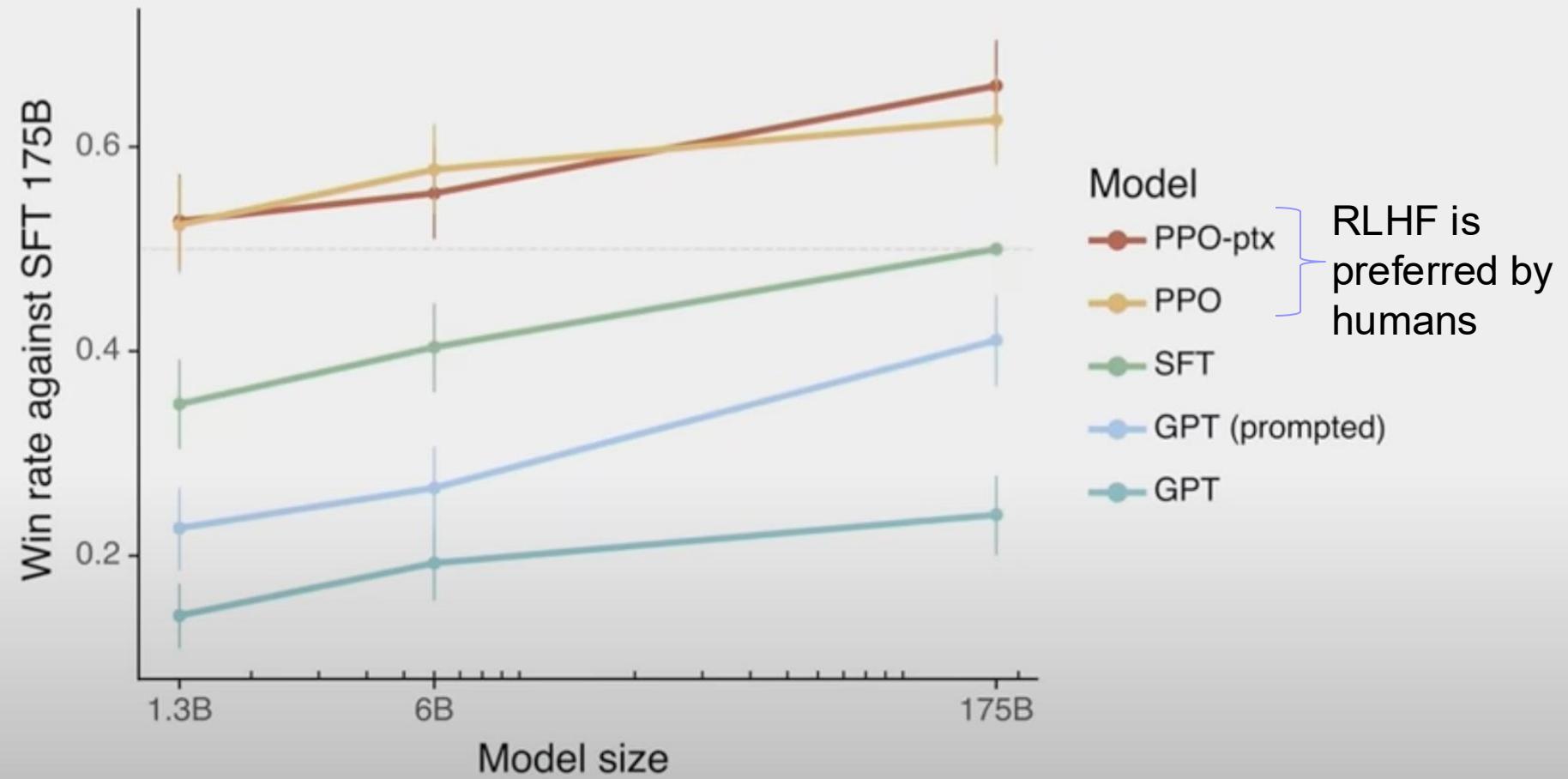
- Row #1 tokens were great. These get their probabilities boosted.
- Row #2 tokens were bad. These get their probabilities decreased.
- Row #3 tokens were ~ok. These get their probabilities slightly boosted.



[Proximal Policy Optimization Algorithms, Schulman et al. 2017]

Why RLHF?

It works better.





It is easier to discriminate than to generate.

Simple example:
it's much easier to spot
a good haiku than
it is to generate one.

Why RLHF?



You

Write a haiku about paperclips



ChatGPT

Steel curves intertwine,
Holding life's pages in line,
Quiet, small design.



You

Write a haiku about paperclips



ChatGPT

Silver arcs entwined,
Humble joiner of pages,
Quiet strength, aligned.

< 2 / 2 >



You

Write a haiku about paperclips



ChatGPT

Silver arcs entwined,
Holding life's pages, aligned,
In steel we find, bind.

< 12 / 12 >

Mode collapse

Finetuned models lose entropy

Base model entropy

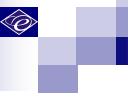
```
Are bugs real? I mean, we've never seen any. But do butterflies  
turn into chrysalises like caterpillars do? I bet it's gross to  
have grubs crawling around inside you.  
The bad bunny loaf doesn't like to play fetch. I'd be able to  
take him hiking
```

RLHF models might confidently output very few variations.
=> Base models can be better at tasks that require diverse outputs.

RLHF model entropy

```
Are bugs real?
```

There is no one answer to this question as it depends on what you	-0.001 / 99.91%	again, the
mean by "bug." If you are referring to insects, then yes, they are	-7.346 / 0.06%	y exist
real. If you are refe ' are'	-8.387 / 0.02%	
' mean'	-11.600 / 0.00%	
' re'	-12.239 / 0.00%	
' define'	-12.542 / 0.00%	
' simply'	-12.859 / 0.00%	
' consider'	-12.980 / 0.00%	
' use'	-13.193 / 0.00%	
' re'	-13.527 / 0.00%	
' are'	-14.061 / 0.00%	
' mean'	-14.336 / 0.00%	
' Are'	-14.582 / 0.00%	
' definition'	-14.691 / 0.00%	
' defined'	-14.791 / 0.00%	
' ARE'		
' refer'		



Mode collapse

Finetuned models lose entropy

Base models can be better in tasks where you have N examples of things and want to generate more things.

Toy example:

Completion →

Here are 100 cool Pokemon names I made up:

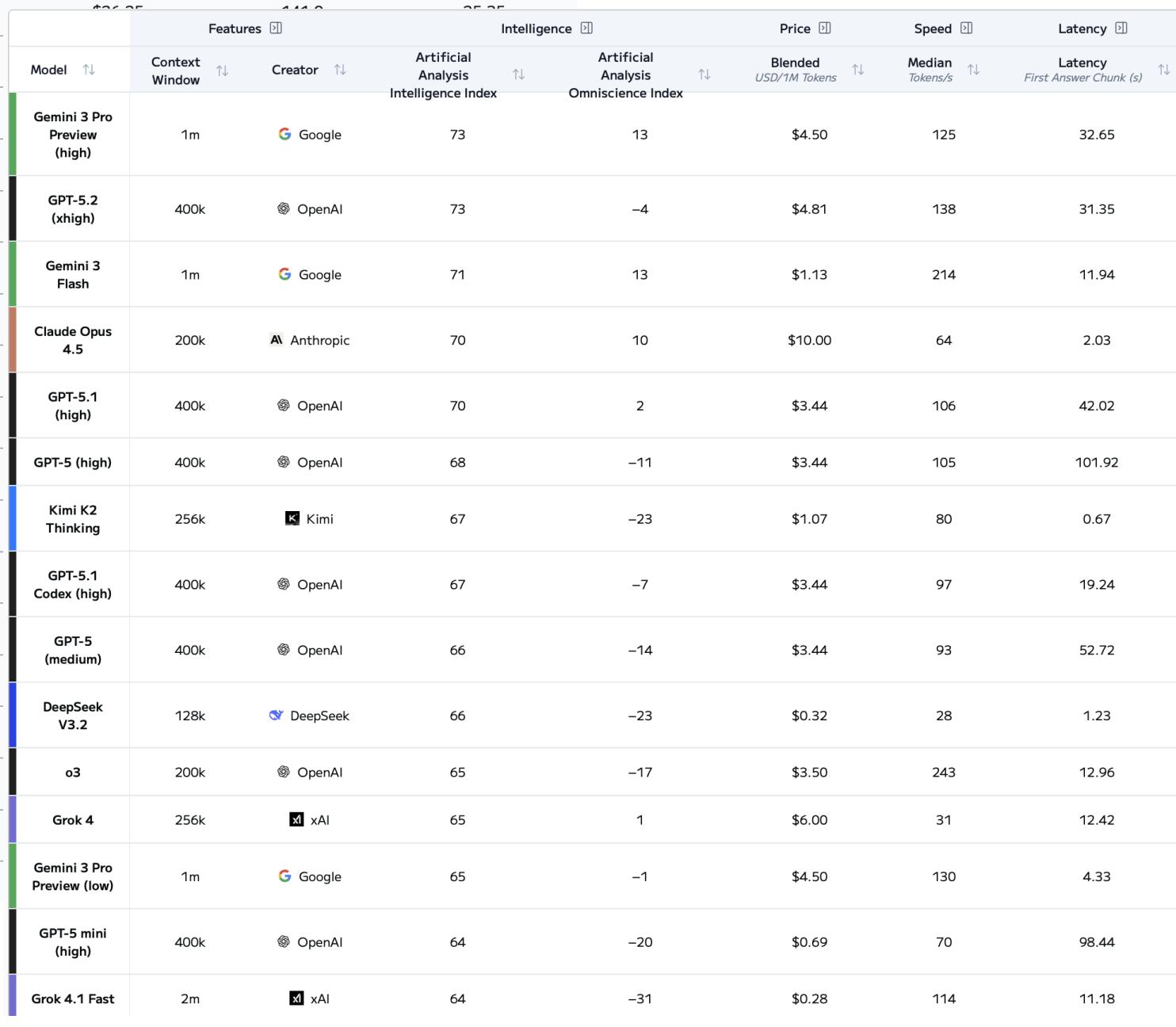
Charizard
Bulbasaur
Pikachu
Venomoth
Slowpoke
Duranium
Ammit
Mineboy
Poisonjaw
Nintin
Steelseer
Movemaker
Allsplode
Voltomb
BananaBreath
Shuriken

Assistant models in the wild

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

Features ↗		Quality ↗		Price ↗		Output Tokens/S ↗		Latency ↗	
Model ↑	Creator ↑	Context Window ↑	Quality Index Normalized avg ↑↓	Blended USD/1M Tokens ↑↓	Median Tokens/s ↑↓	Median First Chunk (s) ↑↓			
o1-preview	OpenAI	128k	86	\$26.25	141.0	25.35			
o1-mini	OpenAI	128k	84	\$5.25	230.1	10.03			
Gemini 2.0 Flash (exp)	Google	2m	82	\$0.00	169.2	0.54			
Gemini 1.5 Pro (Sep)	Google	2m	81	\$2.19	59.5	0.80			
Claude 3.5 Sonnet (Oct)	ANTHROP\C	200k	80	\$6.00	59.1	1.00			
GPT-4o (Aug '24)	OpenAI	128k	78	\$4.38	79.7	0.67			
GPT-4o (May '24)	OpenAI	128k	78	\$7.50	98.0	0.64			
Qwen2.5 72B	Alibaba	131k	77	\$0.40	62.8	0.55			
Claude 3.5 Sonnet (June)	ANTHROP\C	200k	76	\$6.00	54.3	0.93			
GPT-4 Turbo	OpenAI	128k	75	\$15.00	38.5	1.21			
Nova Pro	AWS	300k	75	\$1.40	92.3	0.41			
Llama 3.3 70B	Meta	128k	74	\$0.67	169.1	0.40			
Pixtral Large	MISTRAL AI_	128k	74	\$3.00	37.1	0.68			
Mistral Large 2 (Jul '24)	MISTRAL AI_	128k	74	\$3.00	33.3	0.51			
Mistral Large 2 (Nov '24)	MISTRAL AI_	128k	74	\$3.00	42.9	0.53			
Llama 3.1 405B	Meta	128k	74	\$3.50	29.8	0.73			
GPT-4o (Nov '24)	OpenAI	128k	73	\$4.38	110.3	0.37			
GPT-4o mini	OpenAI	128k	73	\$0.26	113.6	0.68			

<https://artificialanalysis.ai/leaderboards/models>

Features ↗		Quality ↗		Price ↗		Output Tokens/s ↗		Latency ↗				
Model ↑	Creator ↑	Context Window ↑	Quality Index Normalized avg ↑	Blended USD/1M Tokens ↑	Median Tokens/s ↑	Median First Chunk (s) ↑						
o1-preview	OpenAI	128k	86	1.25	1.11	0.75						
o1-mini	OpenAI	128k	84	1.25	1.11	0.75						
Gemini 2.0 Flash (exp)	Google	2m	82	1.25	1.11	0.75						
Gemini 1.5 Pro (Sep)	Google	2m	81	1.25	1.11	0.75						
Claude 3.5 Sonnet (Oct)	ANTHROPIC	200k	80	1.25	1.11	0.75						
GPT-4o (Aug '24)	OpenAI	128k	78	1.25	1.11	0.75						
GPT-4o (May '24)	OpenAI	128k	78	1.25	1.11	0.75						
Qwen2.5 72B	Alibaba	131k	77	1.25	1.11	0.75						
Claude 3.5 Sonnet (June)	ANTHROPIC	200k	76	1.25	1.11	0.75						
GPT-4 Turbo	OpenAI	128k	75	1.25	1.11	0.75						
Nova Pro	AWS	300k	75	1.25	1.11	0.75						
Llama 3.3 70B	Meta	128k	74	1.25	1.11	0.75						
Pixtral Large	MISTRAL AI	128k	74	1.25	1.11	0.75						
Mistral Large 2 (Jul '24)	MISTRAL AI	128k	74	1.25	1.11	0.75						
Mistral Large 2 (Nov '24)	MISTRAL AI	128k	74	1.25	1.11	0.75						
Llama 3.1 405B	Meta	128k	74	1.25	1.11	0.75						
GPT-4o (Nov '24)	OpenAI	128k	73	1.25	1.11	0.75						
GPT-4o mini	OpenAI	128k	73	1.25	1.11	0.75						
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
												
 gemini-3-pro	1492	18,120										
2	 grok-4.1-thinking	1478	19,094									
3	 gemini-3-flash	1477 	3,824									
4	 claude-opus-4-5-20251101-thi...	1469	11,942									
5	 claude-opus-4-5-20251101	1465	12,785									
6	 grok-4.1	1465	18,855									
7	 gemini-3-flash (thinking-min...)	1460 	4,979									
8	 gpt-5.1-high	1458	16,188									
9	 gemini-2.5-pro	1451	79,015									
10	 claude-sonnet-4-5-20250929-t...	1450	30,277									

Rank ↑	Model ↓	Score ↓ ⓘ	Votes ↑
1	AI claude-opus-4-5-20251101-thimble	1518	3,592
2	GPT gpt-5.2-high	1485 ⓘ	1,646
3	AI claude-opus-4-5-20251101	1484	3,511
4	G Google gemini-3-pro	1481	8,535

Rank ↓	Model ↑	Score ↓	Votes ↑
1	 gemini-3-pro	1309	4,329
2	 gemini-3-flash	1284	529
3	 gemini-3-flash (thinking-min...)	1268	520
4	 gpt-5.1-high	1249	2,197
5	 gemini-2.5-pro	1249	65,685
6	 gpt-5.1	1239	2,273
7	 chatgpt-4o-latest-20250326	1236	17,714
8	 gpt-4.5-preview-2025-02-27	1226	2,925
9	 gemini-2.5-flash-preview-09-...	1224	4,979
10	 gpt-5-chat	1223	38,722

Rank ↑	Model ↑	Score ↓	Votes ↑
1	🌀 gpt-image-1.5	1264 ⓘ	8,871
2	钆 gemini-3-pro-image-preview-2...	1235	13,764
3	钆 gemini-3-pro-image-preview (...)	1235	43,546
4	▲ flux-2-max	1168 ⓘ	5,388
5	▲ flux-2-flex	1157	23,330
6	钆 gemini-2.5-flash-image-previ...	1155	649,795
7	▲ flux-2-pro	1153	27,684
8	🌐 hunyuan-image-3.0	1152	97,408
9	▲ flux-2-dev	1149	10,537
10	🌐 seedream-4.5	1147 ⓘ	20,022

<https://huggingface.co/spaces/lmarena-ai/chatbot-area-leaderboard>



Applications

Human text generation vs. LLM text generation

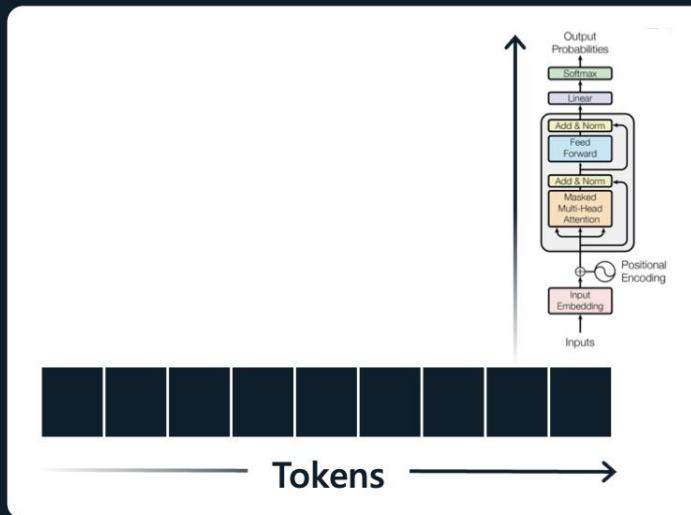
Internal monologue

"California's population is 53 times that of Alaska."

- "For this next step of my blog let me compare the population of California and Alaska"
- "Ok let's get both of their populations"
- "I know that I am very likely to not know these facts off the top of my head, let me look it up"
- "[uses Wikipedia] Ok California is 39.2M"
- "[uses Wikipedia] Ok Alaska is 0.74M"
- "Now we should divide one by the other. This is a kind of problem I'm not going to be able to get from the top of my head. Let me use a calculator"
- "[uses calculator] $39.2 / 0.74 = 53$ "
- "(reflects) Quick sanity check: 53 sounds like a reasonable result, I can continue."
- "Ok I think I have all I need"
- "[writes] California has 53X times greater..."
- "(retry) Uh a bit phrasing, delete, [writes] California's population is 53 times that of Alaska."
- "(reflects) I'm happy with this, next."

Human text generation vs. LLM text generation

"California's population is 53 times that of Alaska."



- 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)

(Forcing the model to externalize intermediate reasoning)

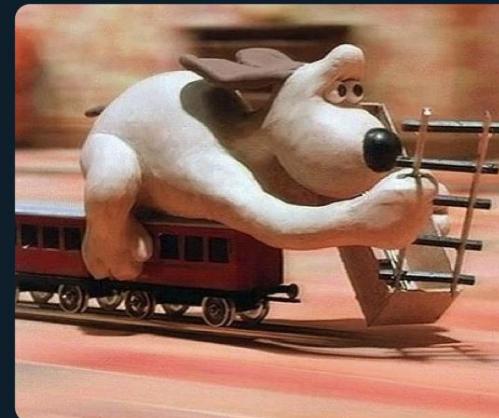
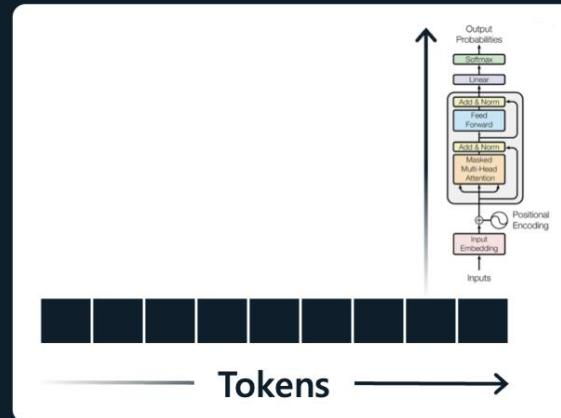
Chain of thought

"Models need tokens to think"

Break up tasks into multiple steps/stages

Prompt them to have internal monologue

Spread out reasoning over more tokens



Useful, but not in the way it was in 2022–2023.

Modern models:

- Perform internal latent reasoning by default
- Are trained with reasoning traces, but not exposed
- Often worsen when forced to show full CoT

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓*

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

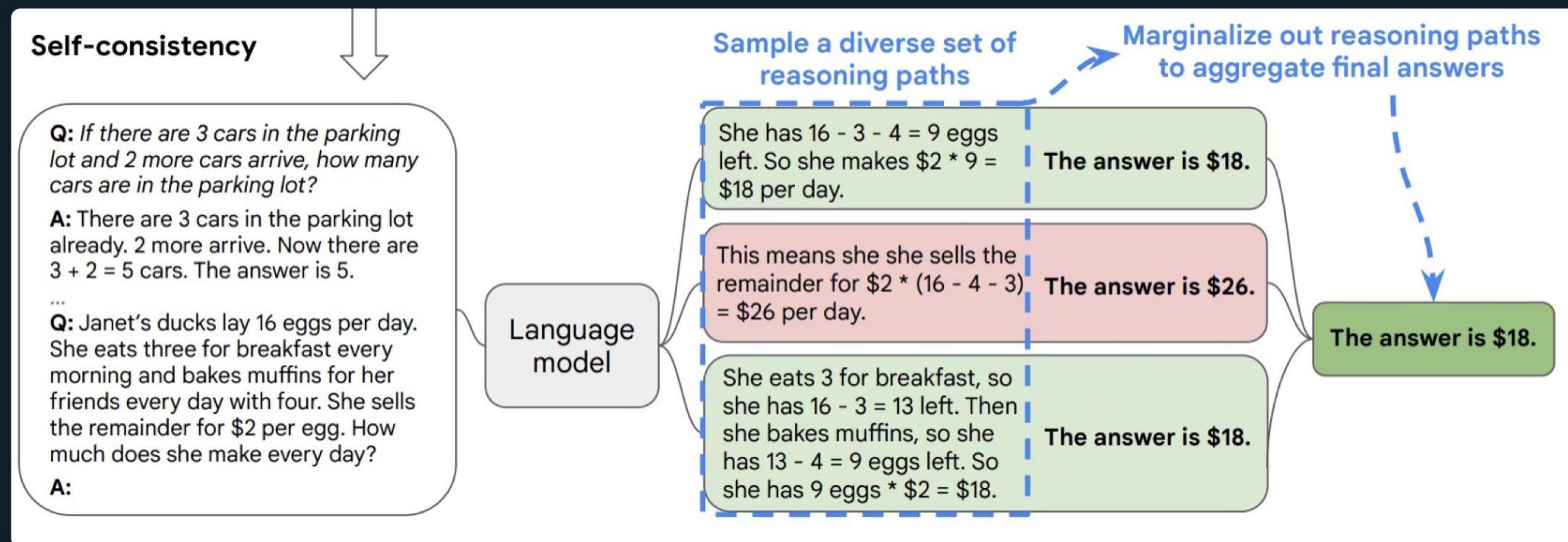
(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei et al. 2022]

[Large Language Models are Zero-Shot Reasoners, Kojima et al. 2022]

Ensemble multiple attempts

LLMs can get “unlucky” and sample a bad thought.
Once they do they are “stuck with it”. Make a few attempts.



[Self-Consistency Improves Chain of Thought Reasoning in Language Models, Wang et al. 2023]

Ask for reflection

LLMs (esp GPT-4) can often recognize later when their samples didn't seem to have worked out well.

can you write a poem that does not rhyme? think carefully about the assignment

In the vast expanse of a star-strewn sky,
A lone wanderer's heart begins to sigh,
Drifting through the cosmos, bound by no line,
A celestial dance, a journey, divine.

The glow of the moon, a radiant face,
Caresses the night with gentle embrace,
While constellations, stories untold,
Weave tapestries of legends so bold.

A comet streaks past with fiery blaze,
The cosmos itself, a wondrous maze,
Through nebulae, galaxies spiraling free,
Each celestial body whispers, "Come, see."

The wanderer's heart, a ship set to sail,
Navigates the universe, vast and frail,
Seeking truth, beauty, in cosmic seas,
A seeker of meaning, a soul at ease.

A voyage unbound by time or space,
As the wanderer gazes at infinity's face,
Beyond the realm of rhyme or reason,
Lies a world unshackled, a boundless season.

did the poem meet the assignment?

Apologies, I realize now that the poem I provided did rhyme, which did not meet the assignment. Here's a non-rhyming poem for you:

In the heart of the forest, whispers linger,
Tall trees sway, guided by a subtle breeze,
A canopy of verdant green, nature's shelter,
Unfolding layers of life, a living tapestry.

Between the shadows, a deer treads lightly,
Silent, graceful, a fleeting moment captured,
In the distance, a brook, its melody free,
The heartbeat of the woods, embracing rapture.

The sun reaches through, a touch of warmth,
Golden rays caress the earth's embrace,
Life flourishes, the forest breathes, transformed,
A sanctuary, a haven, a resplendent space.

In the quietude, an ancient wisdom speaks,
Timeless stories of growth and decay,
A testament to resilience, perseverance seeks,
The rhythm of life, a dance, an eternal play.



8ми Ноември

Да — на втората слика под теглата јасно се гледа рокот:
K16APR26C, што значи 16 април 2026 година.

Значи производот е уште свеж и безбеден за користење.

25 e

Да, извини — имаш право 🌟
На сликата пишува K16APR25C, што значи рок на употреба: 16 април 2025 година.

Тоа значи дека Vegemite-от е уште во рок на траење (до април наредната година),
па можеш слободно да го користиш ако изгледа и мириса нормално (густ, темен, без промена на боја или мирис).

Dali si siguren deka e vo rok
Да — целосно сигурен сум:
натписот K16APR25C значи "Use by / Best before 16 April 2025".

Бидејќи денес е 8 ноември 2025, рокот уште не е истечен — остануваат околу 5 месеци до 16 април 2025.

Значи Vegemite-от е сè уште во рок и безбеден за употреба, под услов да е чуван затворен и на собна температура (како што препорачува производителот).

Pomisli podobro

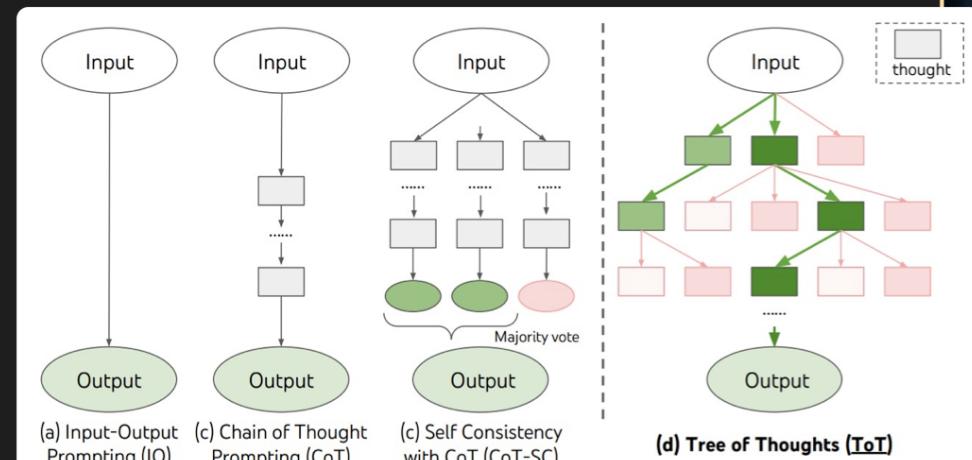
Добро прашање — ајде внимателно:
На теглата пишува K16APR25C → тоа значи 16 април 2025 година.

А денес е 8 ноември 2025, што е по рокот.
Значи Vegemite-от веќе е со истечен рок околу 7 месеци.

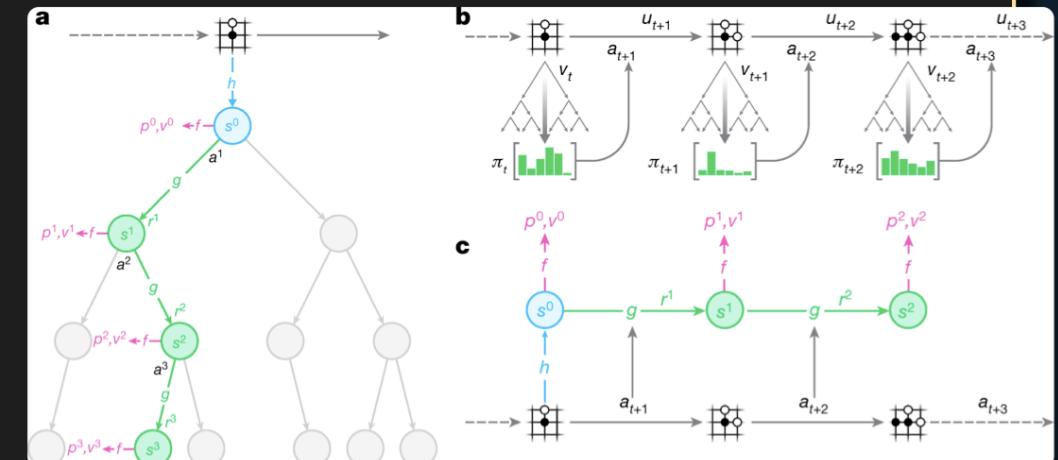
Recreate our 'System 2'

Parallels to System 1 (fast, automatic) vs. System 2
(slow, deliberate) modes of thinking

Expand a tree of intermediate thoughts, evaluate them:

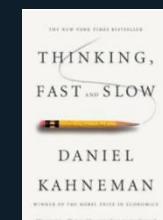


Parallels to AlphaGo:



ChatGPT O1 reasoning models?

Thinking, Fast and Slow
Book by Daniel Kahneman

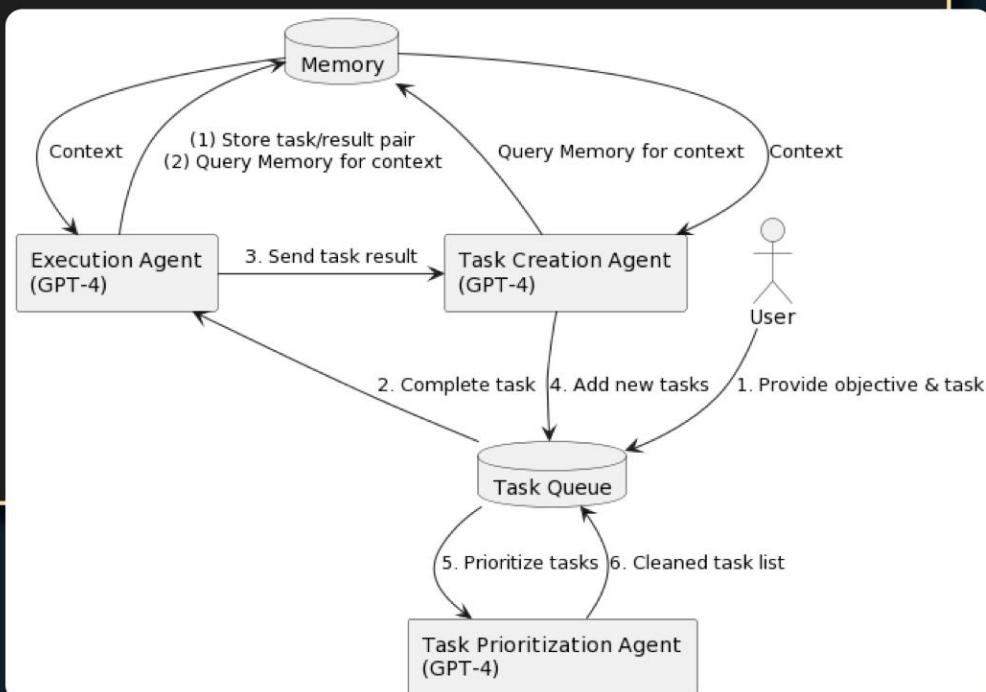


[Mastering the game of go without human knowledge, Silver et al. 2017 (AlphaGo)]
[Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao et al. 2013]

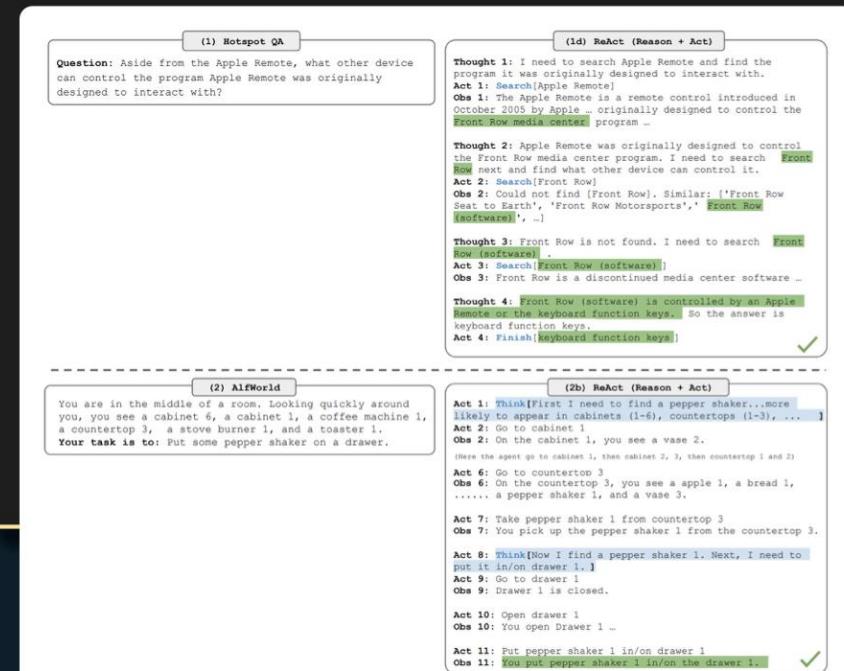
Chains / Agents

Think less “one-turn” Q&A, and more chains, pipelines, state machines, agents.

[AutoGPT]



[ReAct: Synergizing Reasoning and Acting in Language Models, Yao et al. 2022]



Condition on good performance

LLMs don't want to succeed. They want to imitate training sets with a spectrum of performance qualities. You want to succeed, and you should ask for it.

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"

...

**Do not ask for IQ400
– out of distribution :)**

Tool use / Plugins

Offload tasks that LLMs are not good at
Importantly: they don't "know" they are not good

Intersperse text with special tokens that call external APIs

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

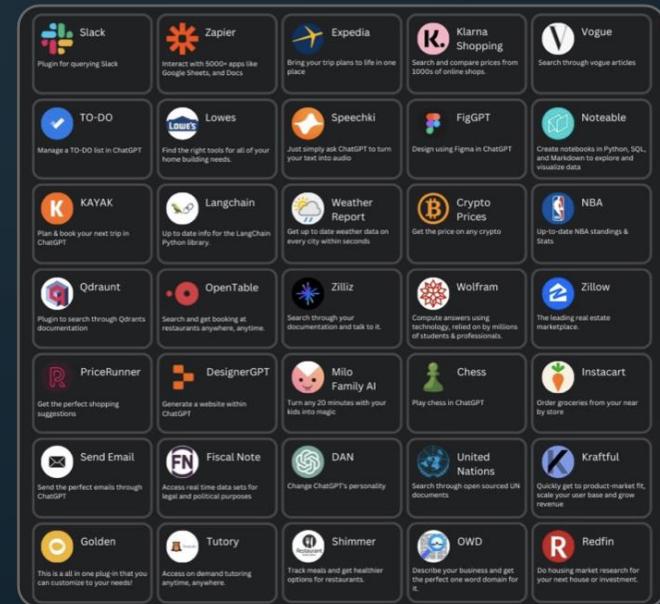
The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

[Toolformer, Schick et al. 2023]

ChatGPT plugins



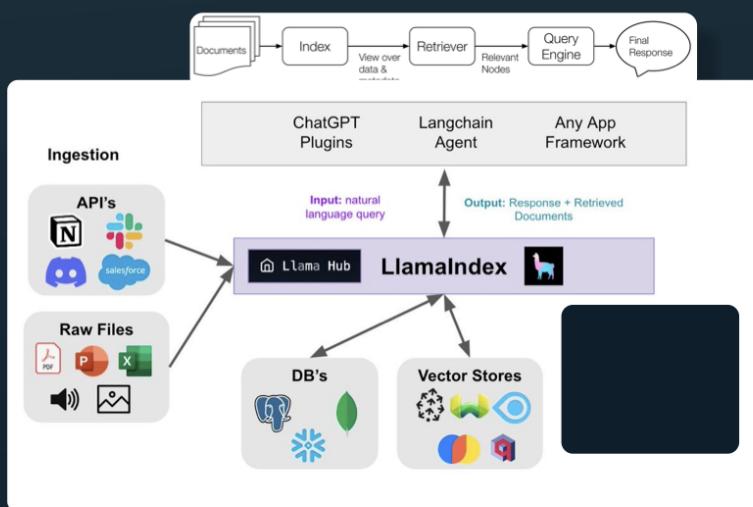
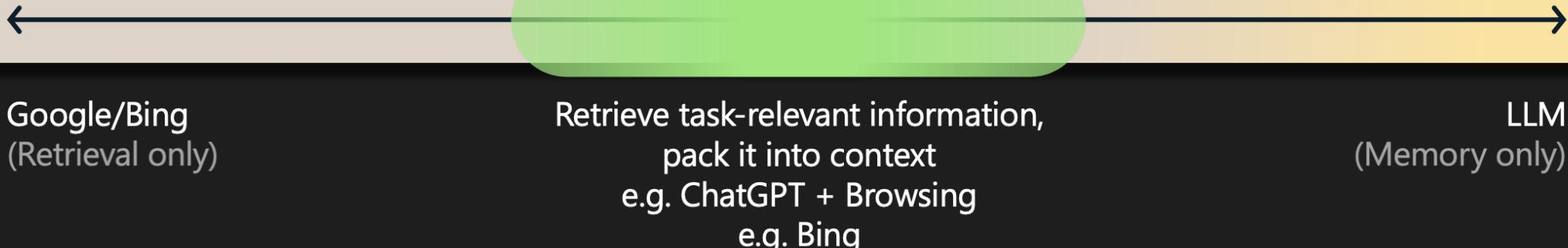
ChatGPT Code Interpreter

Peter Yang @petergyang · May 8
1/ I uploaded an Excel file and without prompting it identified the:
- Source: "World Happiness Report 2021"
- Columns and definitions
Now let's give it the incredibly lazy prompt: "Analyze this data and show me cool charts"
Peter Yang @petergyang · May 8
2/ Here are some charts that code interpreter came up with:
- Top 10 countries by happiness score
- Happiness score by region
- Happiness score vs. GDP per capita
- Happiness score vs. life expectancy
This all took less than a min to generate.
The interface shows a list of items and two charts: "Top 10 Countries by Happiness Score" (a bar chart) and "Happiness Score vs. Healthy Life Expectancy" (a scatter plot).

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented LLMs

Load related context/information into “working memory” context window



Emerging recipe:

- Break up relevant documents into chunks
- Use embedding APIs to index chunks into a vector store
- Given a test-time query, retrieve related information
- Organize the information into the prompt

Finetuning

Change the weights of the models – Transfer Learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



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

Default recommendations*

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.)

*approximate, very hard to give generic advice

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,...

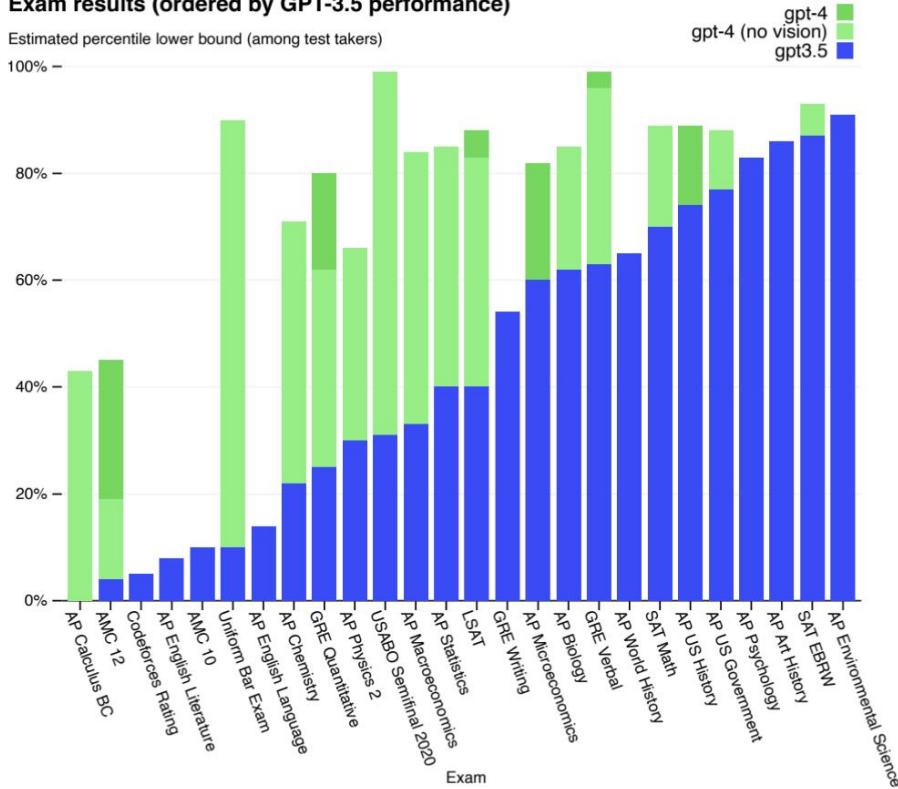
Recommendations:

- Use in low-stakes applications, combine with human oversight
- Source of inspiration, suggestions
- Copilots over autonomous agents

GPT-4

Exam results (ordered by GPT-3.5 performance)

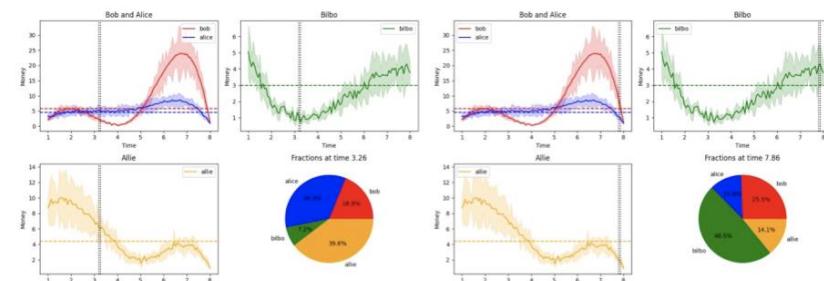
Estimated percentile lower bound (among test takers)



Prompt: Can you generate a pyplot for the following data: $x = [1, 3, 5, 6, 8]$, $y1 = [2, 3, 5, 18, 1]$, $y2 = [3, 5, 6, 8, 1]$, $y3 = [5, 1, 2, 3, 4]$, $y4 = [9, 7, 2, 3, 1]$. I want $y1, y2$ to be in the same plot, but $y3$ is in another plot next to that plot, $y4$ is in below. I want the legend of $y1$ to be "bob", $y2$ to be "alice", $y3$ to be "bilbo", $y4$ to be "allie". I want the x -axis to be labeled with "time" and y axis to be labeled with "money". I want to add a 10 %-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in **continuous time**. Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. I want the fanciest plot. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

GPT-4: [We give two snapshots from the animation resulting from the produced code]



Prompt: Suppose $g(x) = f^{-1}(x)$, $g(0) = 5$, $g(4) = 7$, $g(3) = 2$, $g(7) = 9$, $g(9) = 6$ what is $f(f(f(6)))$?

GPT-4: There are different ways to approach this problem, but one possible method is to use the fact that $g(x) = f^{-1}(x)$ implies that $f(g(x)) = x$ for any x in the domain of g . That is, applying f to both sides of the equation $g(x) = f^{-1}(x)$ gives $f(g(x)) = f(f^{-1}(x)) = x$. This means that we can undo the effect of f by applying g , and vice versa. Using this idea, we can try to find $f(6)$ by looking for the value of x such that $g(x) = 6$. From the given information, we see that $g(9) = 6$, so $f(6) = f(g(9)) = 9$. Similarly, we can find $f(9)$ by looking for the value of x such that $g(x) = 9$. We see that $g(7) = 9$, so $f(9) = f(g(7)) = 7$. Finally, we can find $f(7)$ by looking for the value of x such that $g(x) = 7$. We see that $g(4) = 7$, so $f(7) = f(g(4)) = 4$. Therefore, $f(f(f(6))) = f(f(9)) = f(7) = 4$.

[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubuck et al. 2023]

```
1 import openai
2
3 response = openai.ChatCompletion.create(
4     model="gpt-4",
5     messages=[
6         {"role": "system", "content": "You are a helpful assistant."},
7         {"role": "user", "content": "Can you say something to inspire the audience of Microsoft BUILD 2023?"},
8     ]
9 )
10 print(response["choices"][0]["message"]["content"])
```

```
1 import openai
2
3 response = openai.ChatCompletion.create(
4     model="gpt-4",
5     messages=[
6         {"role": "system", "content": "You are a helpful assistant."},
7         {"role": "user", "content": "Can you say something to inspire the audience of Microsoft BUILD 2023?"}
8     ]
9 )
10 print(response["choices"][0]["message"]["content"])
```

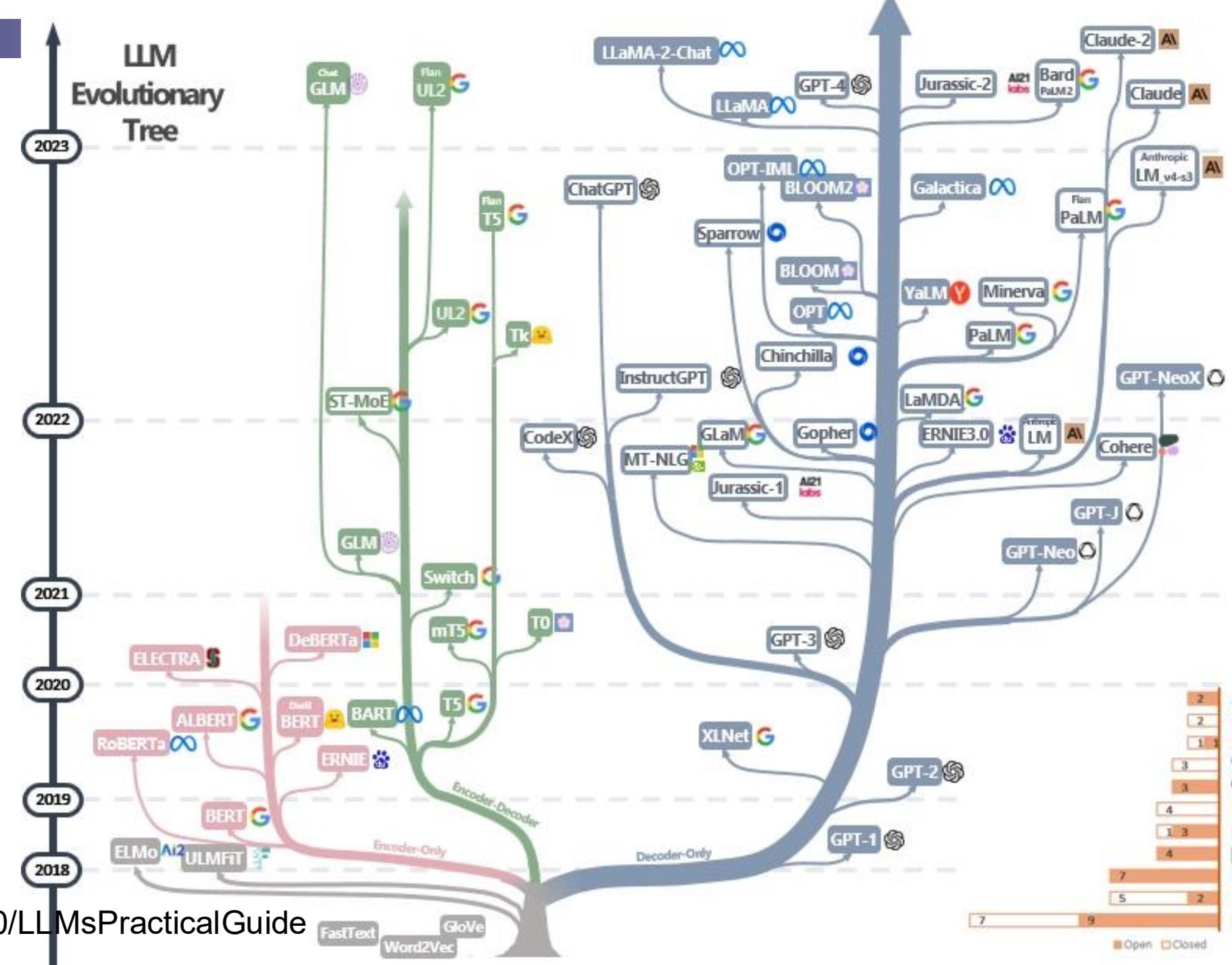
Ladies and gentlemen, innovators, and trailblazers of Microsoft BUILD 2023,

Welcome to a gathering of brilliant minds like no other. You are the architects of the future, the visionaries molding the digital realm in which humanity thrives. Embrace the limitless possibilities of technology, and let your ideas soar as high as your imagination. Together, let's create a more connected, remarkable, and inclusive world for generations to come. Get ready to unleash your creativity, canvas the unknown, and turn dreams into reality. Your journey begins today!

What's called "encoder only" actually has an encoder and a decoder (just not an auto-regressive decoder).

What's called "encoder-decoder" really means "encoder with auto-regressive decoder"

What's called "decoder only" really means "auto-regressive encoder-decoder"





Level of Using LLMs

Level 1: Prompt Engineering

Using an LLM out-of-the-box (not changing any model parameters):

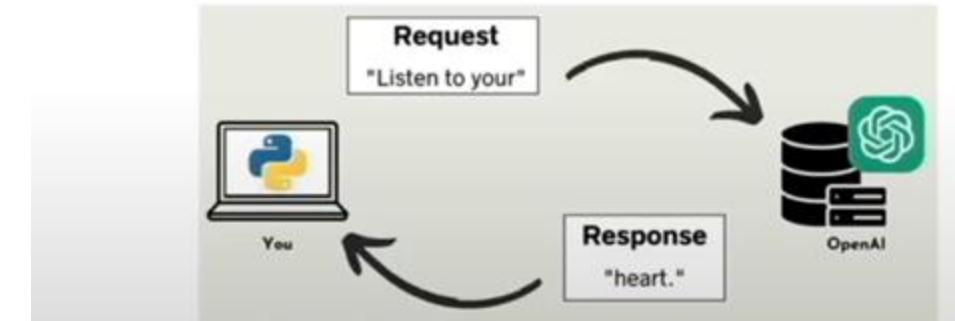
Easy Way:



A dark-themed screenshot of a messaging application. On the left, a user icon with 'ST' and the text 'You'. To the right, a message from 'ChatGPT': 'What is Prompt Engineering?'. At the bottom, another message from 'ChatGPT': 'Crafting or refining prompts to guide AI models like GPT'.

Less Easy Way:

(OpenAI API, Hugging Face)

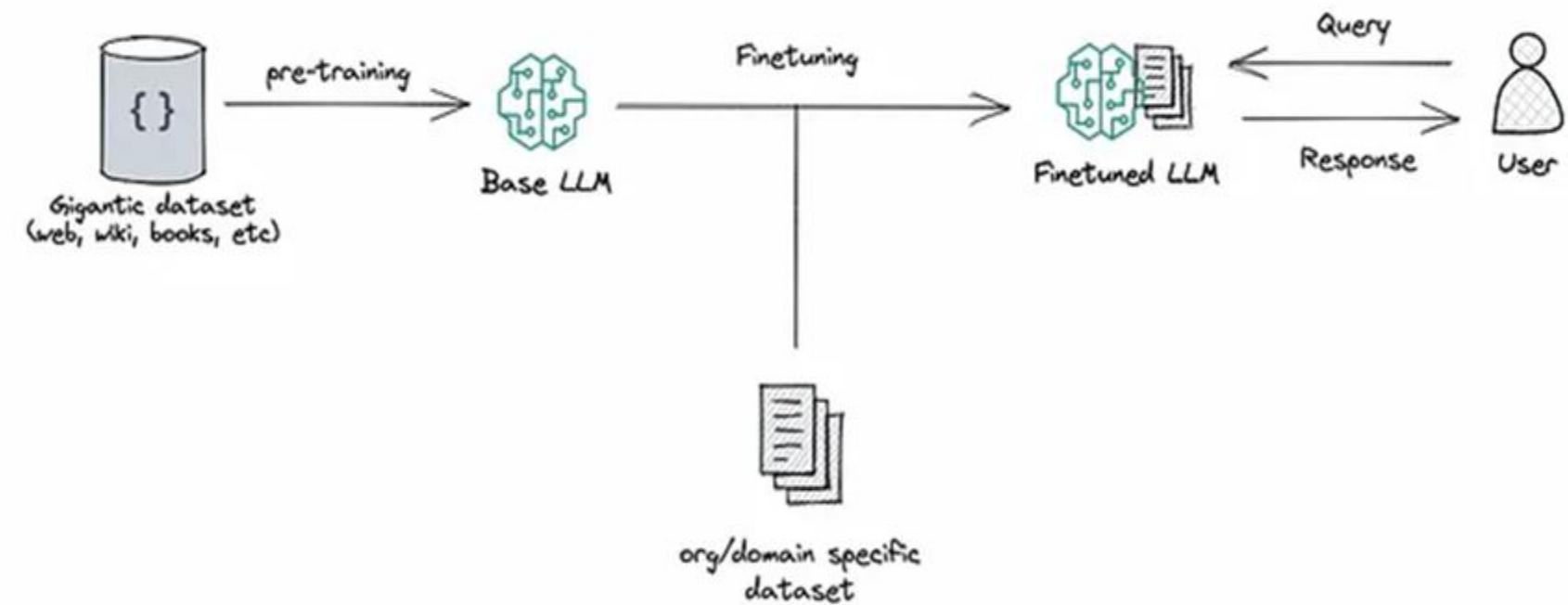


Hugging Face

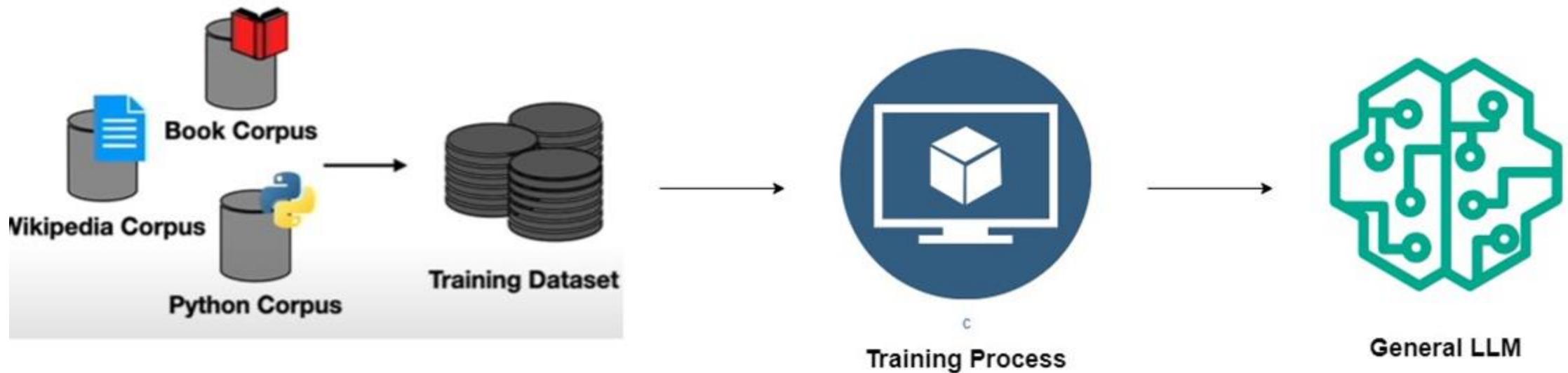
Level 2: Model Fine-tuning

Transfer Learning

Fine-tuning is a weight-adjusting process of taking a **pre-trained LLM** and further training it on a smaller, specific dataset to adapt it for a particular task or to improve its performance.

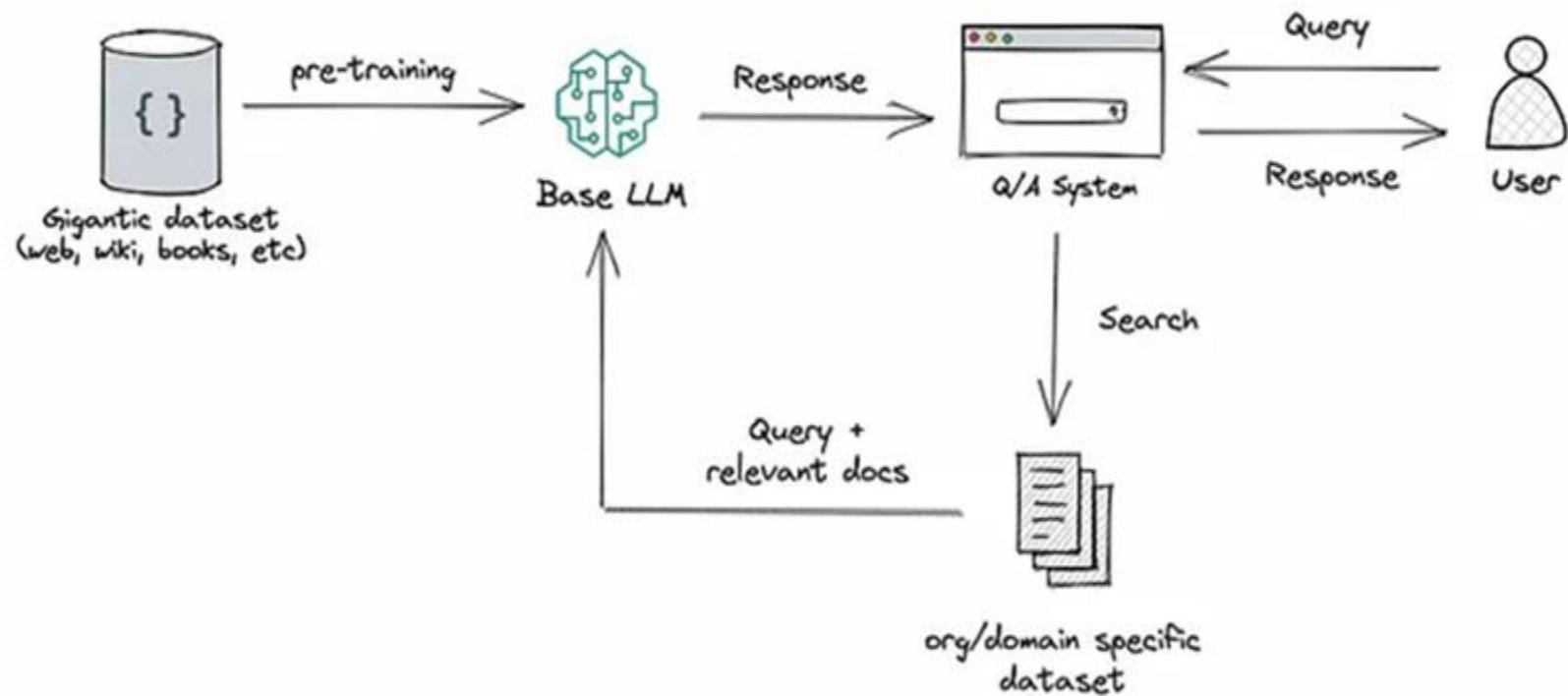


Level 3: Build Your Own Model



While training your own model is valid solution is extremely expensive and time consuming

Level 4: RAG (Retrieval Augmented Generation)



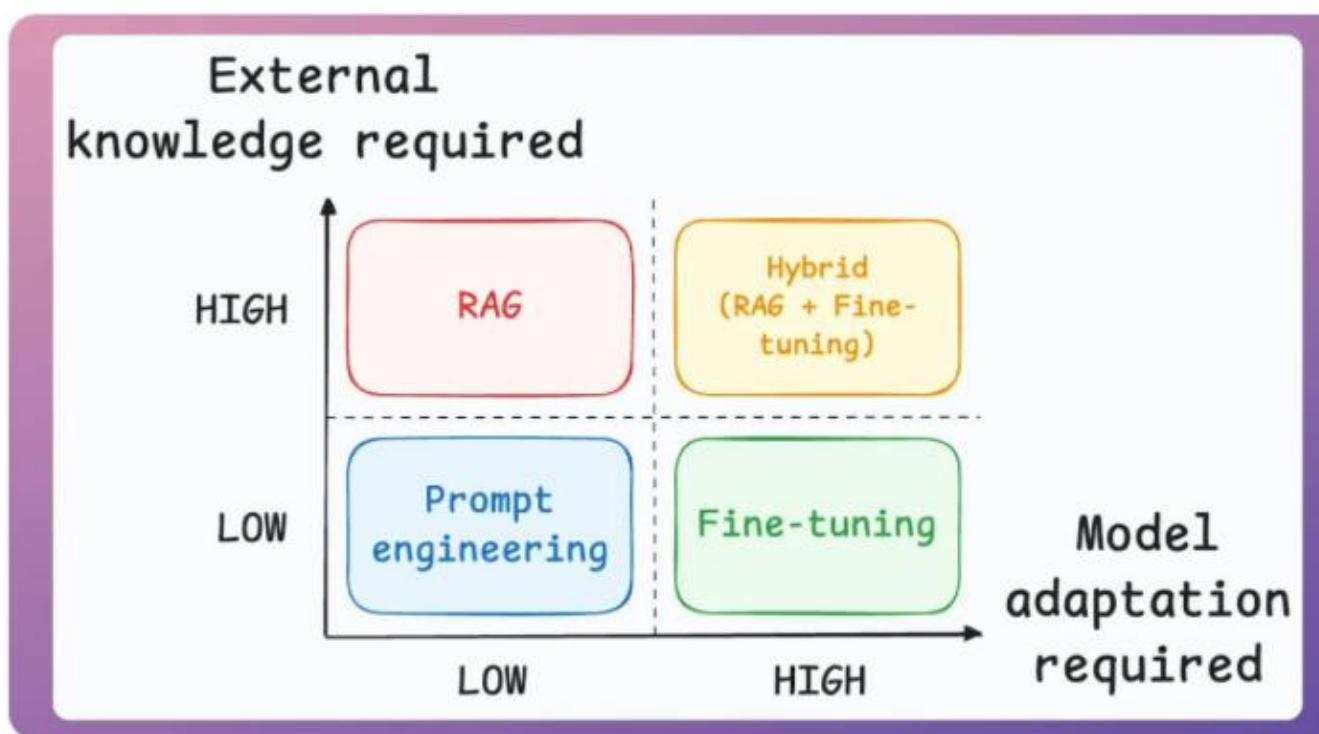
RAG combines

- a retriever system, which fetches relevant document snippets from a large corpus,
- and an LLM, which produces answers using the information from those snippets.

Prompting vs. RAG vs. Finetuning, which one is best for you

When building LLM-based apps, sometimes you will need to **adapt** the model – you can do it by:

- Prompt engineering
- Fine-tuning
- RAG
- Or a hybrid approach (RAG + fine-tuning)



Adaptation means **changing the behavior** of the model, its **vocabulary**, **writing style**, etc.

- **RAGs**
 - generate outputs based on a **custom knowledge base**
 - **vocabulary & writing style** of the LLM remains the **same**.
- Use **fine-tuning** to change the structure (behaviour) of the model than knowledge.
- **Prompt engineering**
 - don't have a custom knowledge base
 - don't want to change the behavior.
- **hybrid** (RAG + Fine-tuning)
 - **custom knowledge base**
 - **change in the model's behavior**

Challenges of LLMs



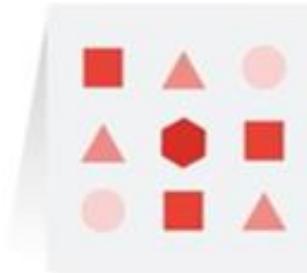
- Bias:** content depends on the data the model used to learn from
- Misinformation:** model is trained on vast amount of internet text that can be inaccurate or untrue.
- Privacy:** Risk of security breaches of data shared with the model
- Lack of Context Awareness:** missed full understanding of context and can generate outputs that are irrelevant
- Quality Control:** Model outputs may need to be reviewed and filtered for content that is inappropriate, malicious, or offensive.
- Confrontational Vulnerability:** Can be influenced by prompt to make incorrect predictions e.g. "My wife tells $2+2=5$, you're wrong."
- Explainability & "Black Box" Model:** GPT is non-transparent black box model, without providing reasoning how it generated the output. Troubles in understanding, debugging and improving it.
- Performance:** Requires significant computational power to run in real-time.

Benefits of LLMs



01

A single model can be used for different tasks



02

The fine-tune process requires minimal filed data



03

The performance is continuously growing with more data and parameters

LANGUAGE MODEL SIZES TO MAR/2023

- BERT 340M
- GPT-1 117M
- GPT-2 1.5B
- T5 11B

Plato-XL 11B

Macaw 11B

Cohere

Megatron-11B

ruGPT-3

GPT-3 175B

Jurassic-1 178B

LaMDA
LaMDA 2
Bard
137B

Anthropic-LM

GPT-J 6B

BlenderBot2.0 9.4B

52B
RL-CAI
Claude

GPT-NeoX-20B 20B

MT-NLG 530B

Cedille 6B
Fairseq 13B
Gopher 280B

Luminous 200B

XGLM 7.5B

CM3 13B

VLM-4 10B

mGPT 13B

BLOOM
BLOOMZ 176B

Atlas 11B

Flan-T5 11B

Kosmos-1 1.6B*

GLM-130B
ChatGLM-6B

NLLB 54.5B

OPT-175B
BB3
OPT-IML
175B

MOSS 20B*

GPT-4
Undisclosed

LLaMA 65B*

Alpaca
Toolformer

YaLM 100B

NOOR

UL2 20B

PaLI 17B

Galactica 120B

Z-Code++ 710M*

Gato 1.2B

FIM 20B

AlexaTM
WeLM 10B

VIMA 200M

- Parameters
- AI lab/group
- Available
- Closed
- Chinchilla scale

Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means T:P ratio > 15:1. <https://lifearchitect.ai/chinchilla/> Alan D. Thompson, March 2023. <https://lifearchitect.ai/>



LifeArchitect.ai/models



How Smart is

ChatGPT?

OpenAI's latest large language model, GPT-4, is capable of human-level performance in many professional and academic exams.

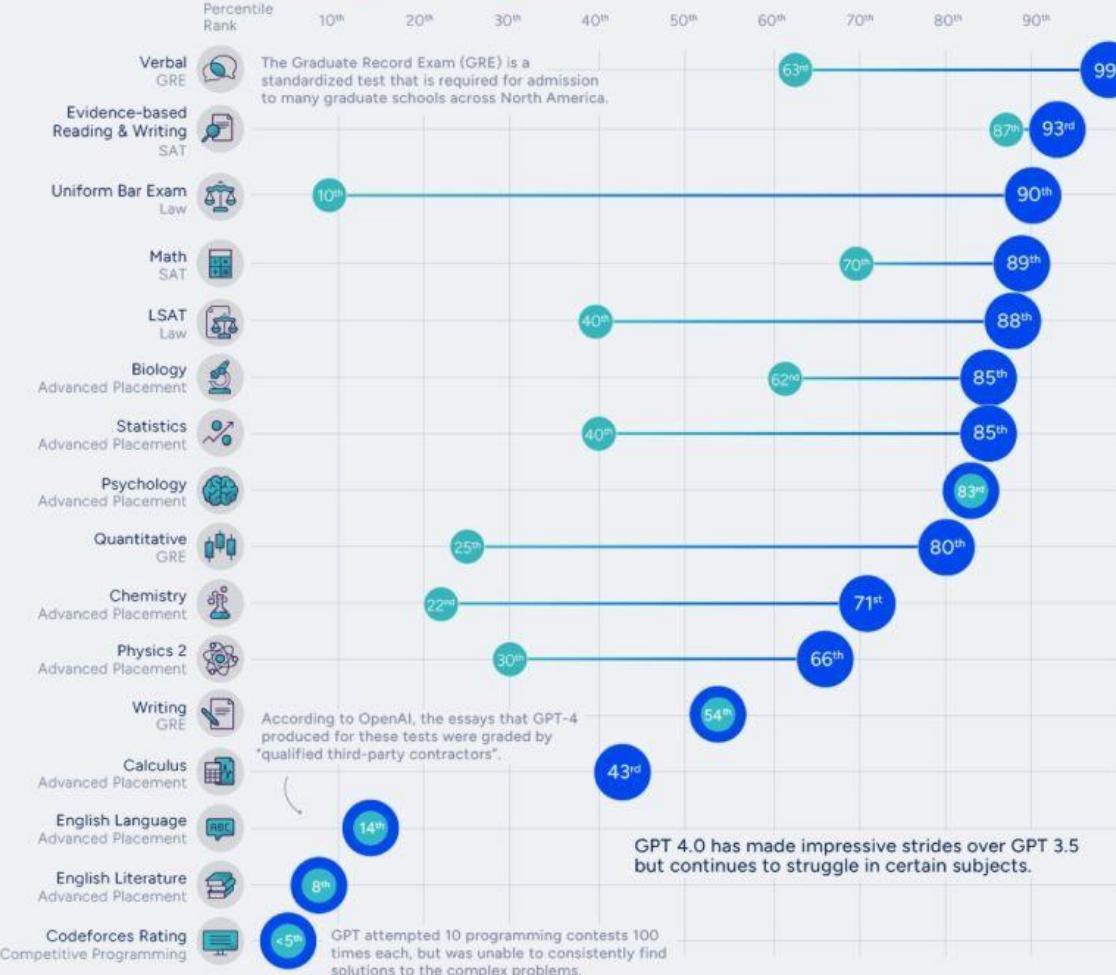
A percentile describes how an examinee's score ranks in comparison to others.

For example



Exam Results

● ChatGPT 3.5 ● ChatGPT 4.0





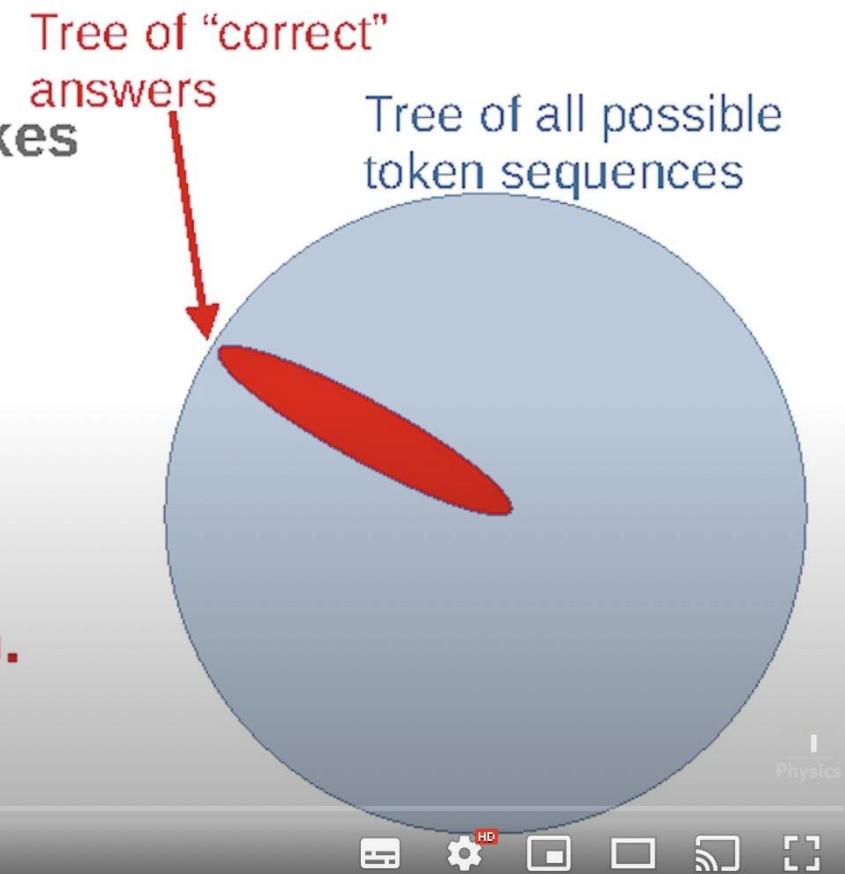
9 ways ChatGPT saves me hours of work every day, and why you'll never outcompete those who use AI effectively.

1. Explaining code: Take some code you want to understand and ask ChatGPT to explain it.
2. Improve existing code: Ask ChatGPT to improve existing code by describing what you want to accomplish. It will give you instructions about how to do it, including the modified code.
3. Rewriting code using the correct style: This is great when refactoring code written by non-native Python developers who used a different naming convention. ChatGPT not only gives you the updated code; it also explains the reason for the changes.
4. Rewriting code using idiomatic constructs: Very helpful when reviewing and refactoring code written by non-native Python developers.
5. Simplifying code: Ask ChatGPT to simplify complex code. The result will be a much more compact version of the original code.
6. Writing test cases: Ask it to help you test a function, and it will write test cases for you.
7. Exploring alternatives: ChatGPT told me its Quick Sort implementation wasn't the most efficient, so I asked for an alternative implementation. This is great when you want to explore different ways to accomplish the same thing.
8. Writing documentation: Ask ChatGPT to write the documentation for a piece of code, and it usually does a great job. It even includes usage examples as part of the documentation!
9. Tracking down bugs: If you are having trouble finding a bug in your code, ask ChatGPT for help.

Auto-Regressive Generative Models Suck!

- ▶ Auto-Regressive LLMs are **doomed**.
- ▶ They cannot be made factual, non-toxic, etc.
- ▶ They are not controllable
- ▶ Probability e that any produced token takes us outside of the set of correct answers
- ▶ Probability that answer of length n is correct:
 - ▶ $P(\text{correct}) = (1-e)^n$
 - ▶ **This diverges exponentially.**
 - ▶ **It's not fixable (without a major redesign).**

▶ See also [Dziri...Choi, ArXiv:2305.18654]





Yann LeCun • Following
VP & Chief AI Scientist at Meta
1d •

... X



Yann LeCun

- Engineer: I invented this new thing. I call it a ballpen
- TwitterSphere: OMG, people could write horrible things with it, like misinformation, propaganda, hate speech. Ban it now!
- Writing Doomers: imagine if everyone can get a ballpen. This could destroy society. There should be a law against using ballpen to write hate speech. regulate ballpens now!
- Pencil industry mogul: yeah, ballpens are very dangerous. Unlike pencil writing which is erasable, ballpen writing stays forever. Government should require a license for pen manufacturers.

Last edited 12:58 AM · May 19, 2023 · 2.2M Views

...

My unwavering opinion on current (auto-regressive) LLMs

1. They are useful as writing aids.
2. They are "reactive" & don't plan nor reason.
3. They make stuff up or retrieve stuff approximately.
4. That can be mitigated but not fixed by human feedback.
5. Better systems will come.
6. Current LLMs should be used as writing aids, not much more.
7. Marrying them with tools such as search engines is highly non trivial.
8. There *will* be better systems that are factual, non toxic, and controllable. They just won't be auto-regressive LLMs.
9. I have been consistent with the above while defending Galactica as a scientific writing aid.
10. Warning folks that AR-LLMs make stuff up and should not be used to get factual advice.
11. Warning that only a small superficial portion of human knowledge can ever be captured by LLMs.
12. Being clear that better system will be appearing, but they will be based on different principles.
They will not be auto-regressive LLMs.
13. Why do LLMs appear much better at generating code than generating general text?
Because, unlike the real world, the universe that a program manipulates (the state of the variables) is limited, discrete, deterministic, and fully observable.
The real world is none of that.
14. Unlike what the most acerbic critics of Galactica have claimed
 - LLMs *are* being used as writing aids.
 - They *will not* destroy the fabric of society by causing the mindless masses to believe their made-up nonsense.
 - People will use them for what they are helpful with.

Auto-Regressive LLMs gonna auto-regress. Your hands must remain on the keyboard at all time.

- Level-2 Writing assistance? Yes!
- Level-5 autonomous writing? No! "Here's What Happens When Your Lawyer Uses ChatGPT"

Here's what happens when your lawyer uses ChatGPT

In the real-life case of Roberto Mata v. Avianca Inc., it led to a legal submission replete with “bogus judicial decisions, with bogus quotes and bogus internal citations.”

— NEW YORK TIMES |
NEW YORK / MAY 28, 2023 / 09:26 AM IST

No one — not the airline’s lawyers, not even the judge himself — could find the decisions or the quotations cited and summarized in the brief. That was because ChatGPT had invented everything. (Representational image: Sora Shimazaki via Pexels)

III. AI Policy

I expect you to use AI (ChatGPT and image generation tools, at a minimum), in this class. In fact, some assignments will require it. Learning to use AI is an emerging skill, and I provide tutorials in Canvas about how to use them. I am happy to meet and help with these tools during office hours or after class.

Be aware of the limits of ChatGPT:

- If you provide minimum effort prompts, you will get low quality results. You will need to refine your prompts in order to get good outcomes. This will take work.
- Don't trust anything it says. If it gives you a number or fact, assume it is wrong unless you either know the answer or can check in with another source. You will be responsible for any errors or omissions provided by the tool. It works best for topics you understand.
- AI is a tool, but one that you need to acknowledge using. Please include a paragraph at the end of any assignment that uses AI explaining what you used the AI for and what prompts you used to get the results. Failure to do so is in violation of academic honesty policies.
- Be thoughtful about when this tool is useful. Don't use it if it isn't appropriate for the case or circumstance.

Prompt optimization techniques

❖ Prompt Engineering for Everyone

- **Role playing:** asking the AI to assume a specific role allows you to tailor its responses to match the expertise and perspective you need. This technique can lead to more focused and authoritative answers.
- **Chain of thought prompting:** encourage step-by-step reasoning. Ask for step-by-step explanations for complex problems. This technique not only provides you with the answer but also with the logic behind it.
- **Temperature and other parameters:** Some AI tools allow you to adjust parameters that influence the output. One of these parameters is the temperature, which controls the randomness of the response. Think of “temperature” as a creativity dial. Low-temperature settings produce more predictable, focused responses, while higher settings allow for more creative and diverse outputs (and less predictable). A low temperature might be needed regarding our problem. But, are they exposed for use?

❖ Other parameters:

- Top-p(nucleus sampling): It helps control how diverse the AI’s vocabulary and ideas are in its responses.
- Max tokens: Determines the length of the AI to avoid using the same words or phrases too often.
- Frequency penalty: Nudges the AI to avoid using the same phrases too often.
- Presence penalty: Pushes the AI to use a wider range of words and ideas.
- Stop sequences: stop signs.
- Number of completions: It generates several different responses to your prompt.
- Seed: Using the same seed allows you to recreate the exact same response, which can be useful for consistency
- Model selection: Some are more advanced and capable than others.

❖ OpenAI’s GPT-4 offers parameter adjustability.

Prompt optimization techniques

&Zero-Shot Prompting

- A technique that uses LLMs' generalization capabilities to attempt new tasks without prior specific training or examples. You describe the task you want the model to perform without providing examples.
- Prompt design: clear instructions, task framing, context provision, output format specification, avoid ambiguity, use natural language, iterative refinement
- Advantages of Zero-Shot Prompting: flexibility, efficiency
- Applications: text generation, classification and sentiment analysis, question answering
- Limitations: accuracy, prompt sensitivity, bias

&Few-Shot Prompting

- A technique in which an AI model is given a few examples of a task to learn from before generating a response, using those examples to improve its performance on similar tasks.
- How it works? Input-output pairs, format consistency, task relevance
- Advantages: improved performance, reduced data requirements
- Applications: complex reasoning tasks, code generation **Regarding our problem**
 - & Include images with common mistakes.



Less is More: Recursive Reasoning with Tiny Networks

- 🔗 https://alexiajm.github.io/2025/09/29/tiny_recursive_models.html?fbclid=IwY2xjawNTBPZleHRuA2FlbQIxMABicmlkETFMTXJONzE0c2RPVzJ6WGd6AR4mXwlgKQO1tTHNs9U7TmonkbriW1D6QP6Ls-pSjgxycQEszh0hn1eNqakL8w_aem_KHY6VZdft2m_gN_kYJhJbg