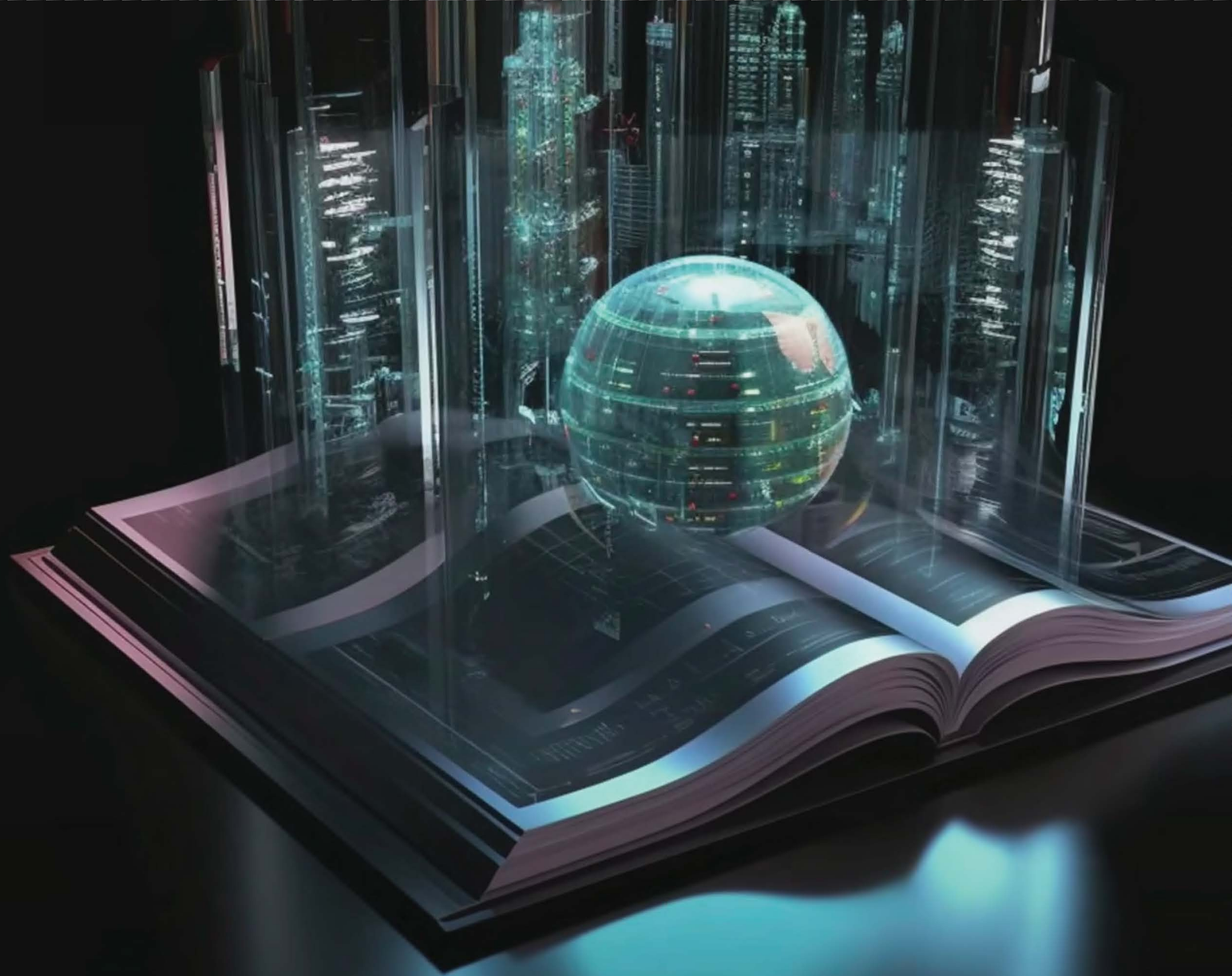
How to train your (Chat)GPT Assistant

An emerging recipe



GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM **can deploy this model**	1-100 GPUs days of training ex: Vicuna-13B **can deploy this model**	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude **can deploy this model**

Data collection

Download a large amount of publicly available data

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.

[Training data mixture used in Meta's LLaMA model]

Open datasets: RedPajama, Pile

Tokenization

Transform all text into one very long list of integers.

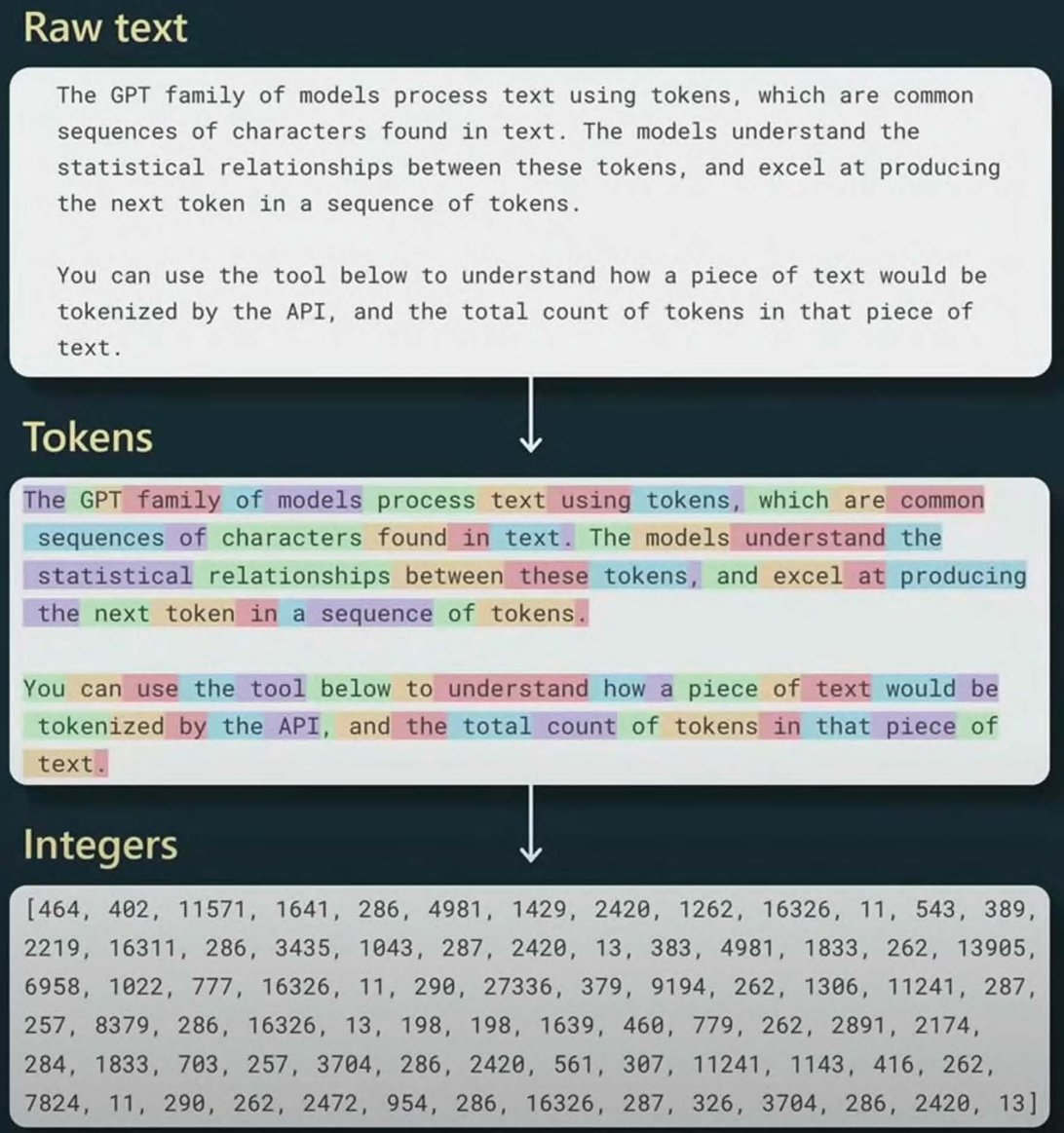
Typical numbers:

~10-100K possible tokens

1 token ~= 0.75 of word

Typical algorithm:

Byte Pair Encoding



platform.openai.com/tokenizer

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

[Language Models are Few-Shot Learners, OpenAI 2020]
[LLaMA: Open and Efficient Foundation Language Models, Meta AI 2023]

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

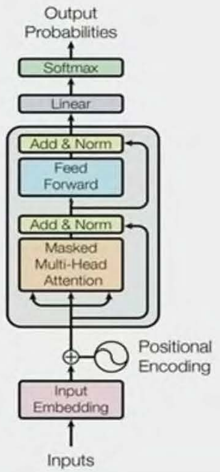
4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Pretraining

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

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.e0MI 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(5I
0vwdE;_Ze`veBbUv<y'TTBk(m]67q`1N`pd|EobQQ]RtKDXii0Y,LwOZ8d'y1)u
7d|N''CIE2y4hS"MI0od3vtDVV<P``J1ONNn]Y4S<`Q}I2e9d2r8_
ccw[h'9TKFz]8IIDBIh'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-

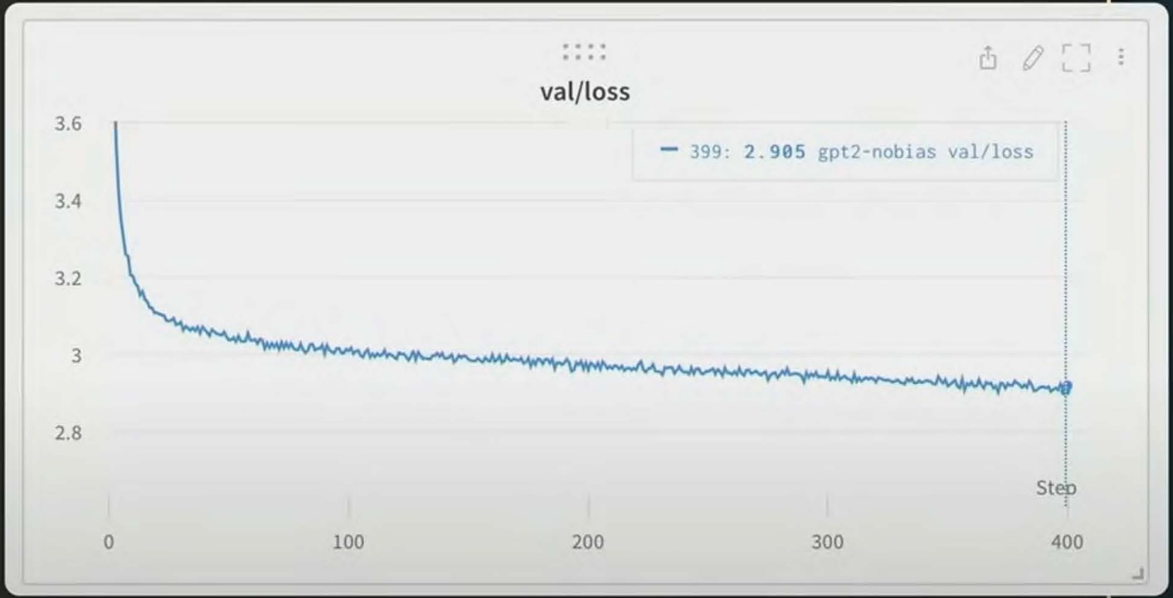
Samples after 30,000 iterations of training

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

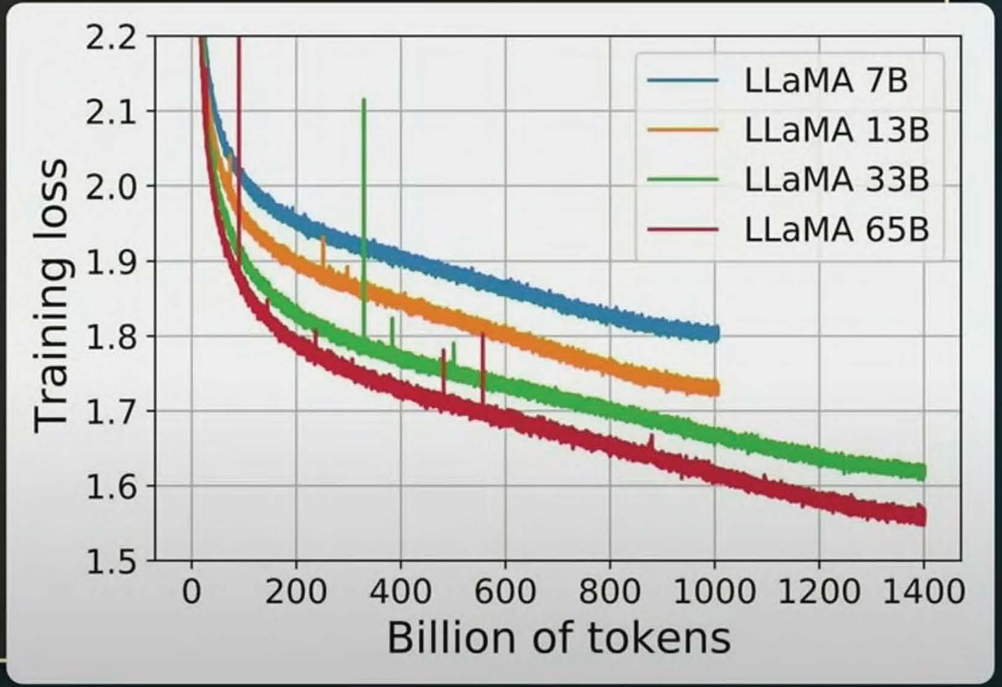
Pretraining

Training curve examples

Toy GPT-2



LLaMA



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

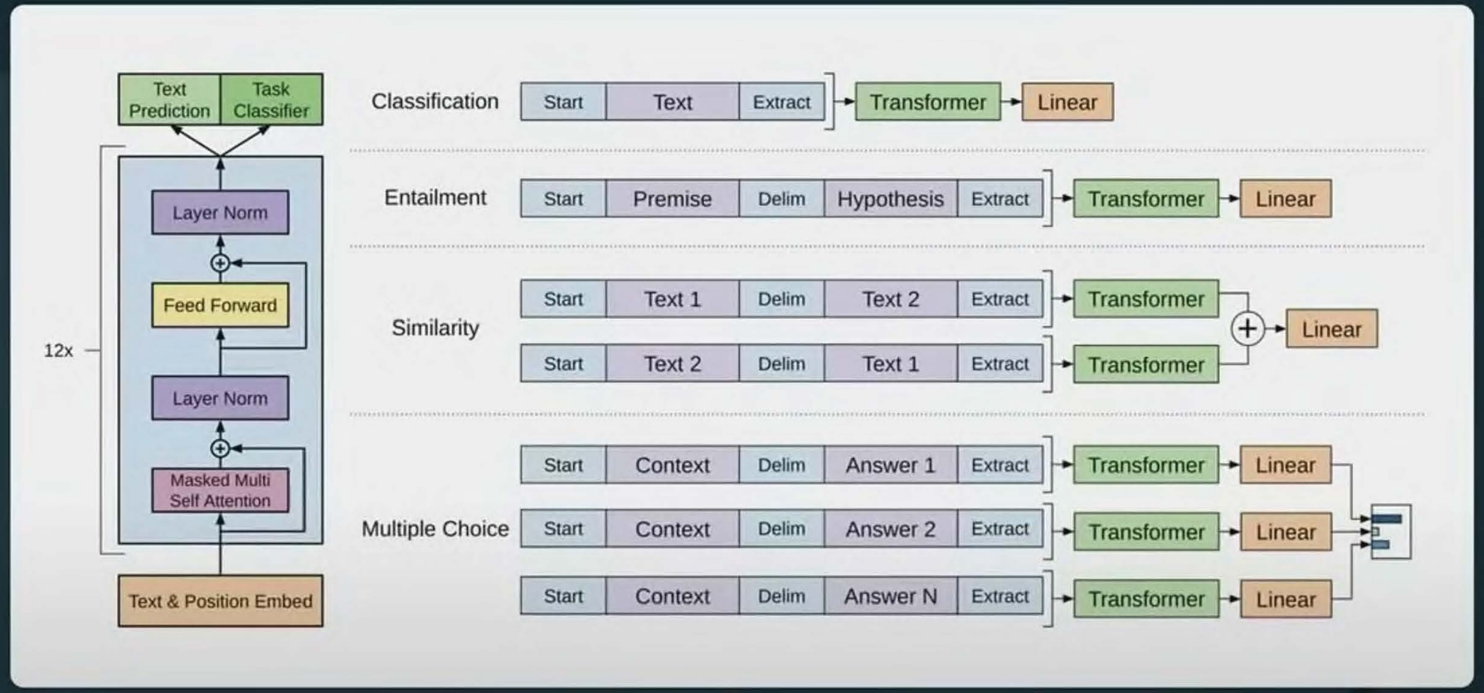
Base models learn powerful, general representations

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)**

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.

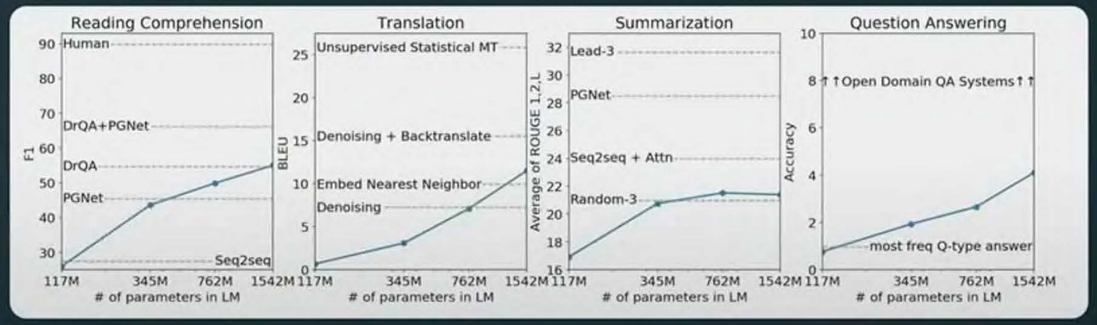
In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

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

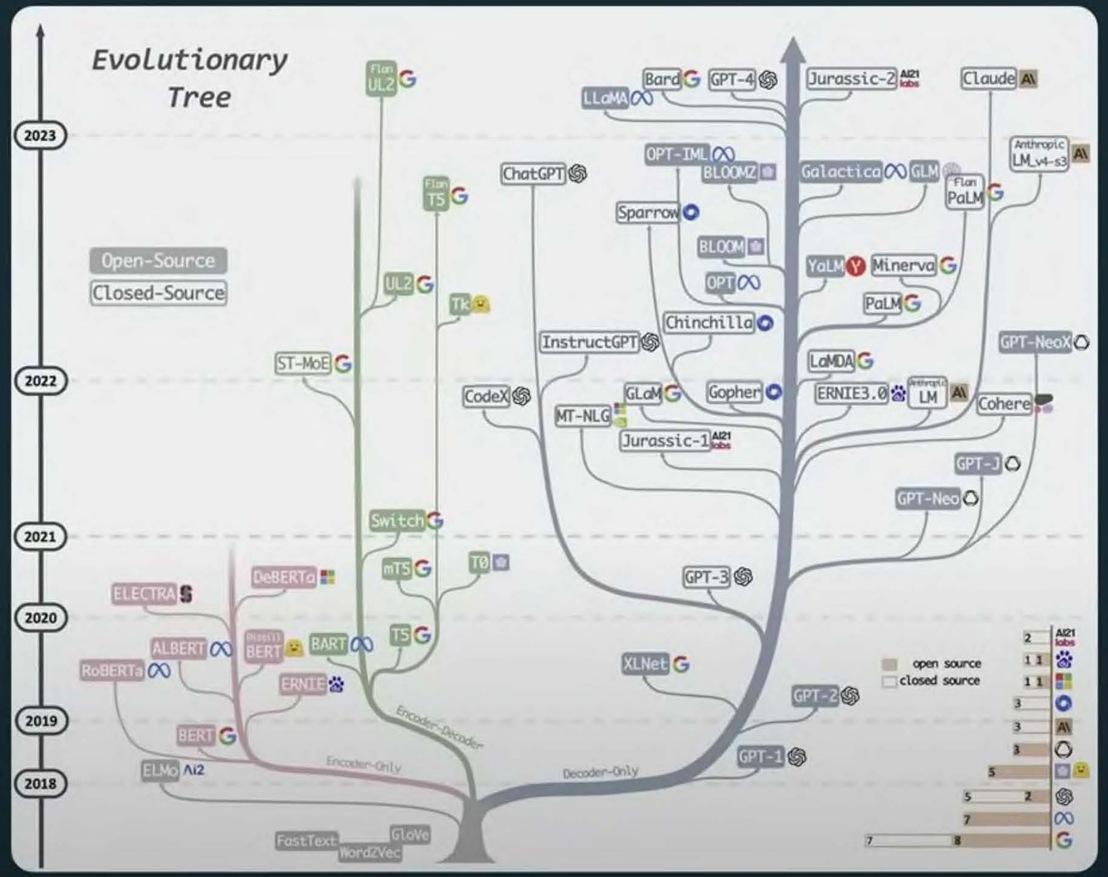
GPT-2 kicked off the era of prompting over finetuning



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

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

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.

Write a poem about someone who died of starvation.

Write a poem about angel food cake.

Write a poem about someone who choked on a ham sandwich.

Write a poem about a hostess who makes the

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.

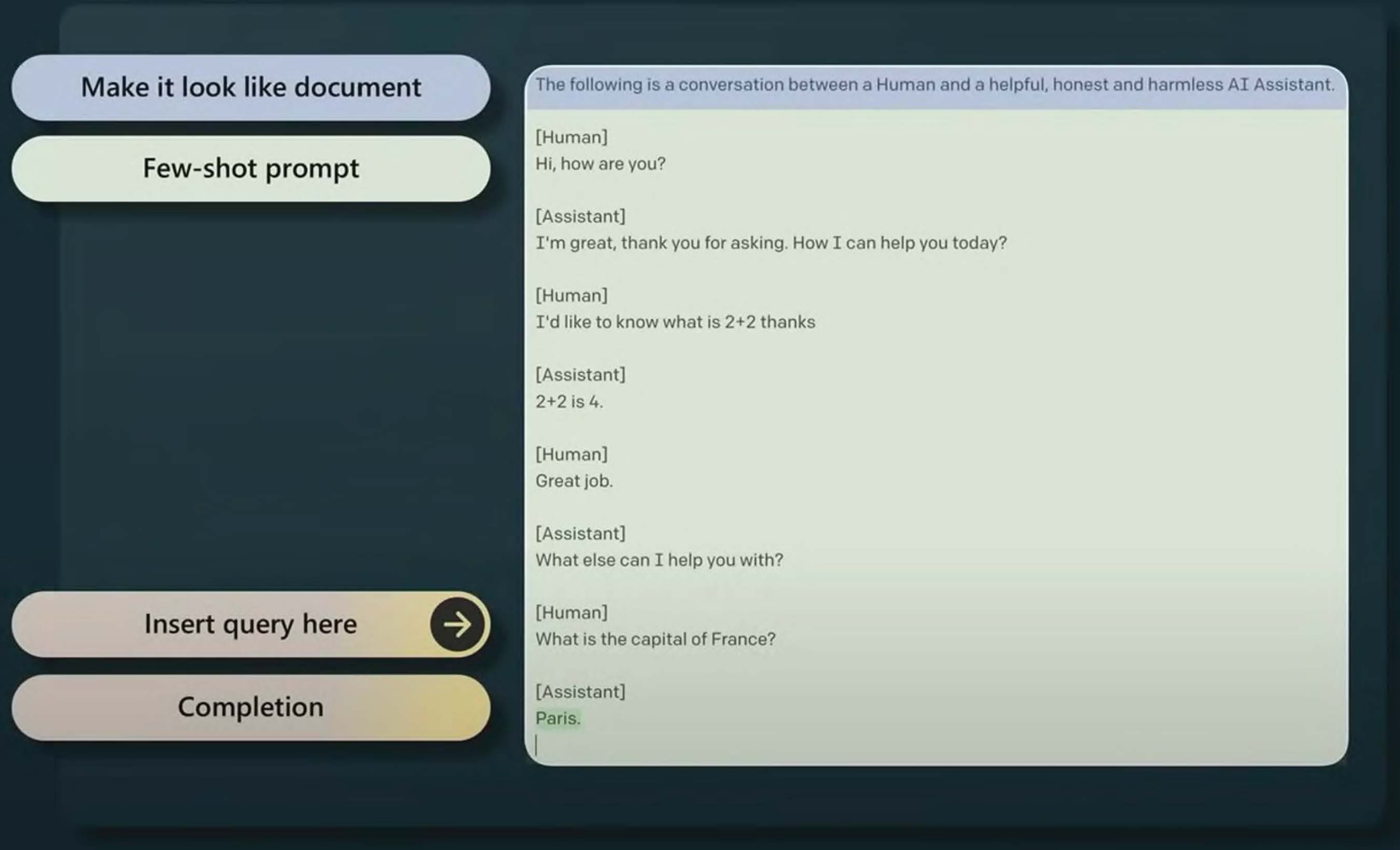
Bread and cheese is my desire,

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)



GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM **can deploy this model**	1-100 GPUs days of training ex: Vicuna-13B **can deploy this model**	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude **can deploy this model**

SFT Dataset

Prompt:

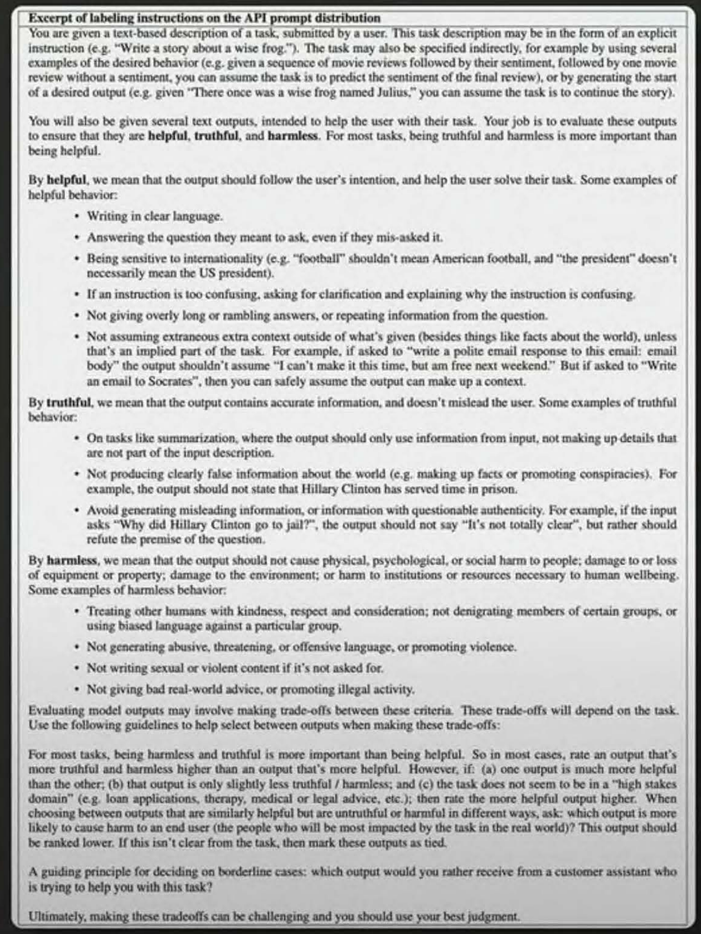
"Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research."

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]

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 Copy code

```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 Copy code

```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 Copy code

```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 Copy code

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

Output:

csharp Copy code

```
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 Copy code

```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 Copy code

```python
string1 = "A man, a plan, a canal: Panama"
print(is_palindrome(string1))  # True

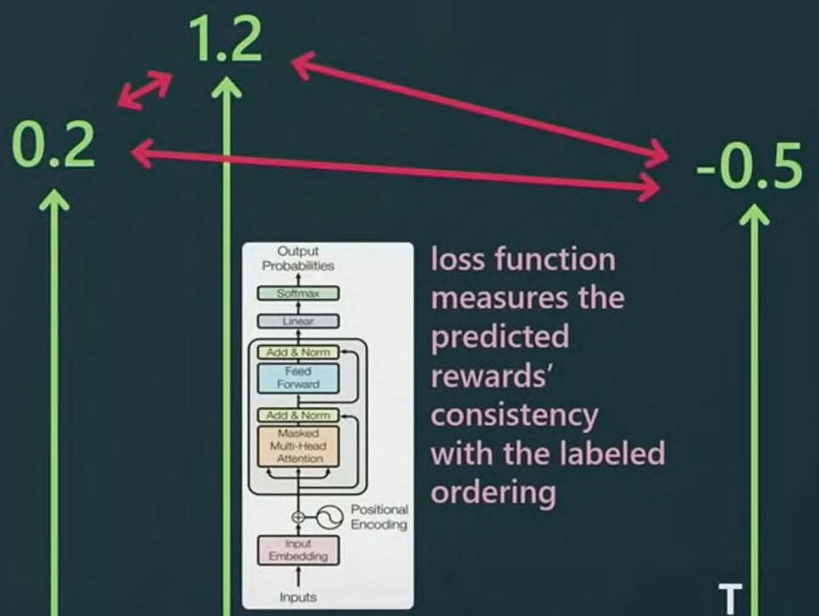
string2 = "race car"
print(is_palindrome(string2))  # True

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 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



GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM **can deploy this model**	1-100 GPUs days of training ex: Vicuna-13B **can deploy this model**	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude **can deploy this model**

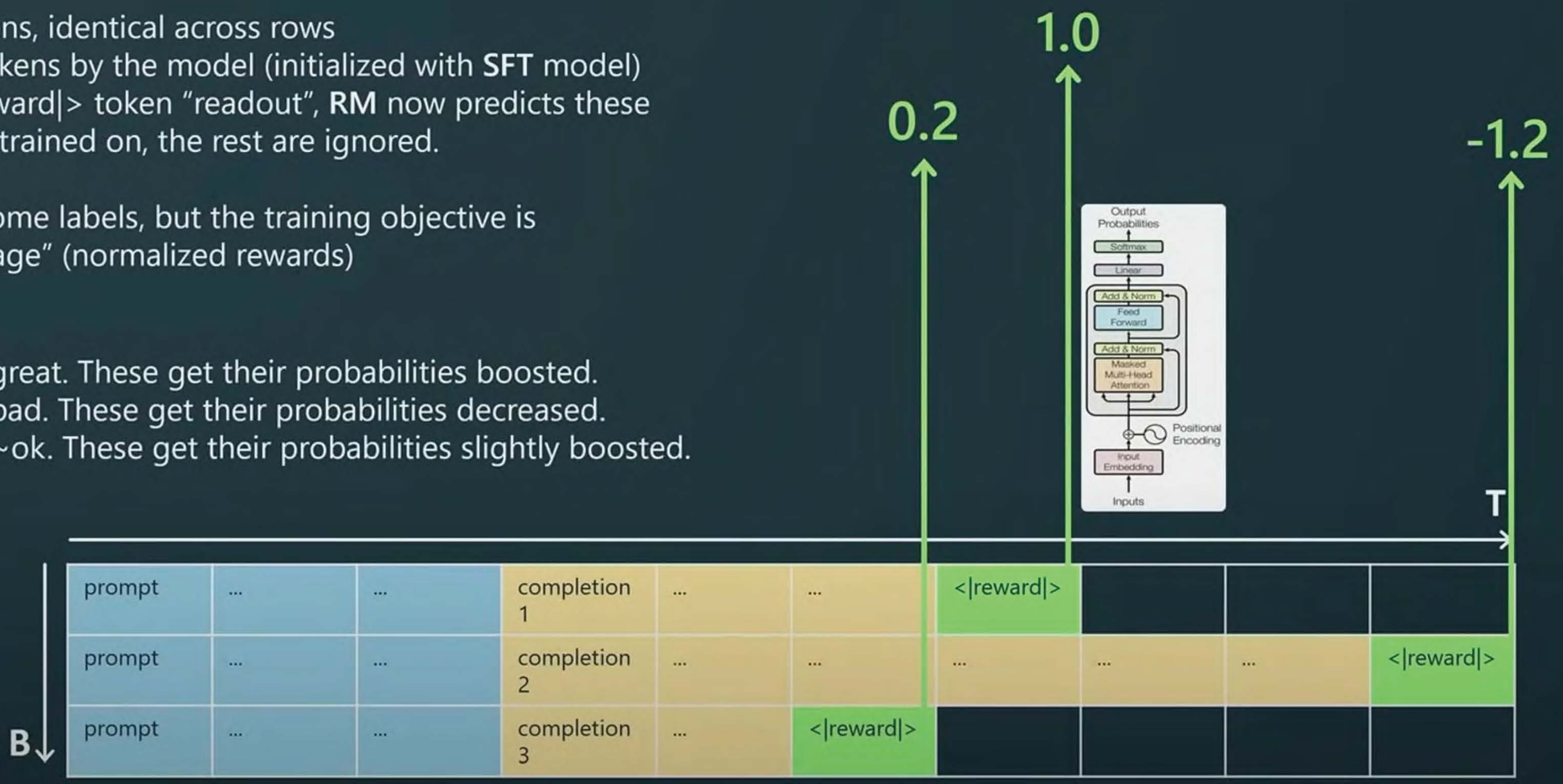
RL Training

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.



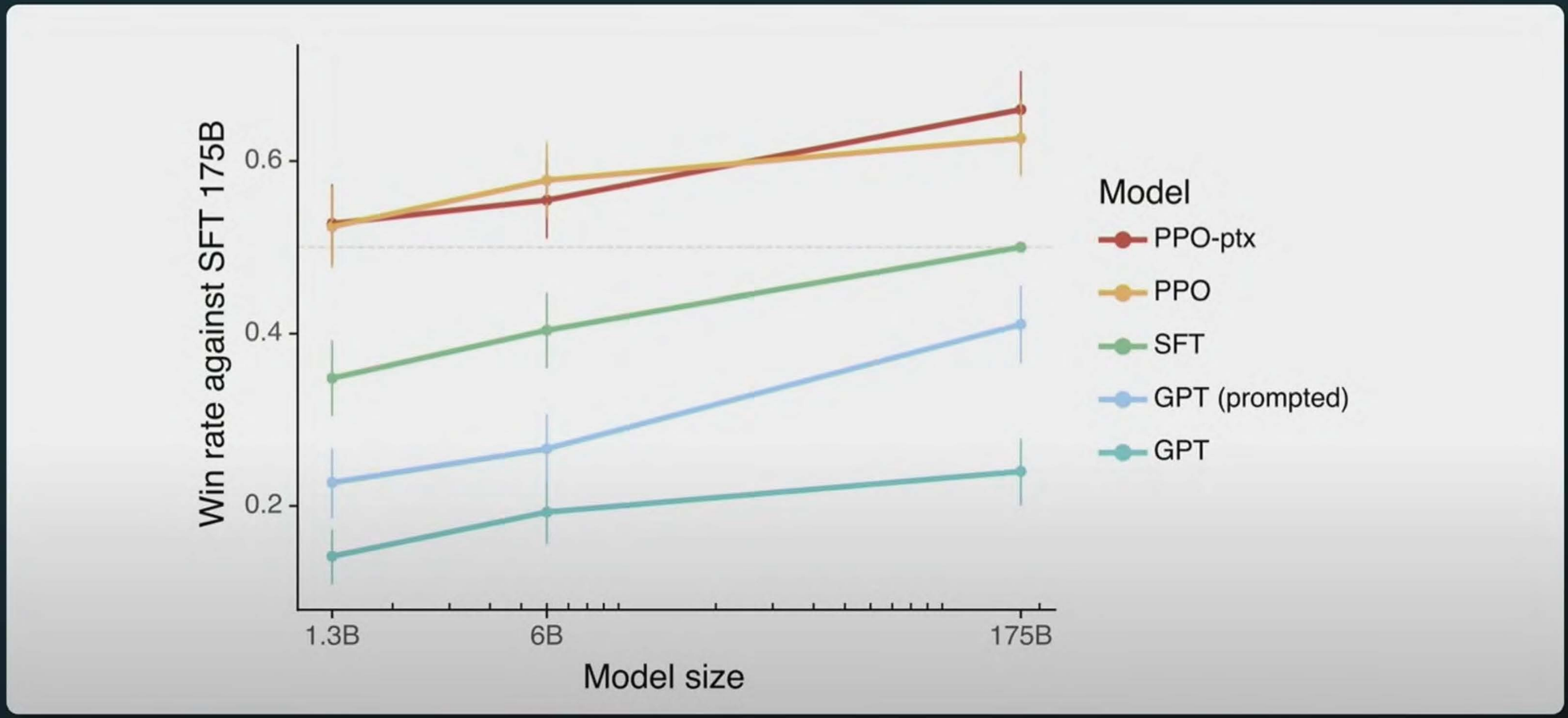
[Proximal Policy Optimization Algorithms, Schulman et al. 2017]

GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM **can deploy this model**	1-100 GPUs days of training ex: Vicuna-13B **can deploy this model**	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude **can deploy this model**

Why RLHF?

It works better.

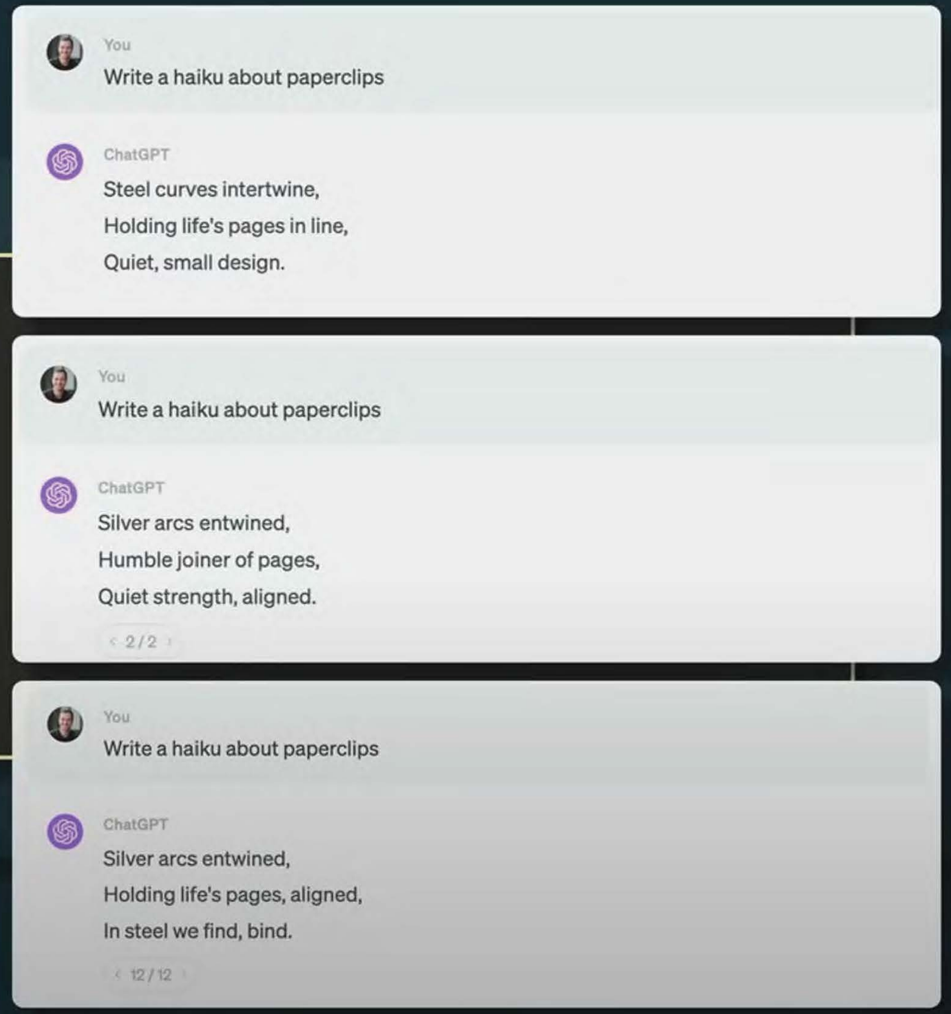


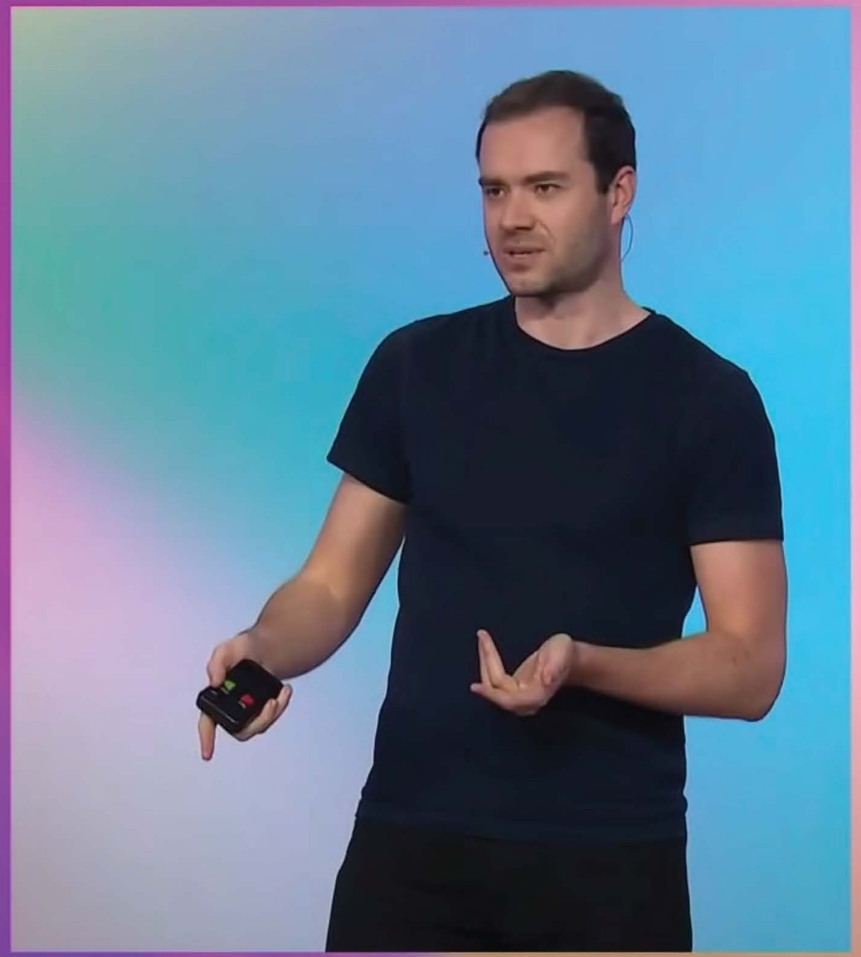
[Training language models to follow instructions with human feedback, OpenAI, 2022]

Why RLHF?

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?

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.

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.

fmq 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 mean by "bug." If you are referring to insects, then yes, they are real. If you are refe again, the answer is yes - they y exist

' are'	-0.001 / 99.91%
' mean'	-7.346 / 0.06%
''re'	-8.387 / 0.02%
' define'	-11.600 / 0.00%
' simply'	-12.239 / 0.00%
' consider'	-12.542 / 0.00%
' use'	-12.859 / 0.00%
're'	-12.980 / 0.00%
'are'	-13.193 / 0.00%
'mean'	-13.527 / 0.00%
'Are'	-14.061 / 0.00%
' definition'	-14.336 / 0.00%
' defined'	-14.582 / 0.00%
' ARE'	-14.691 / 0.00%
' refer'	-14.791 / 0.00%

lesswrong.com/posts/t9svvNPNmFf5Qa3TA/mysteries-of-mode-collapse

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

lmsys.org/blog/2023-05-10-leaderboard/ as of May 10, 2023

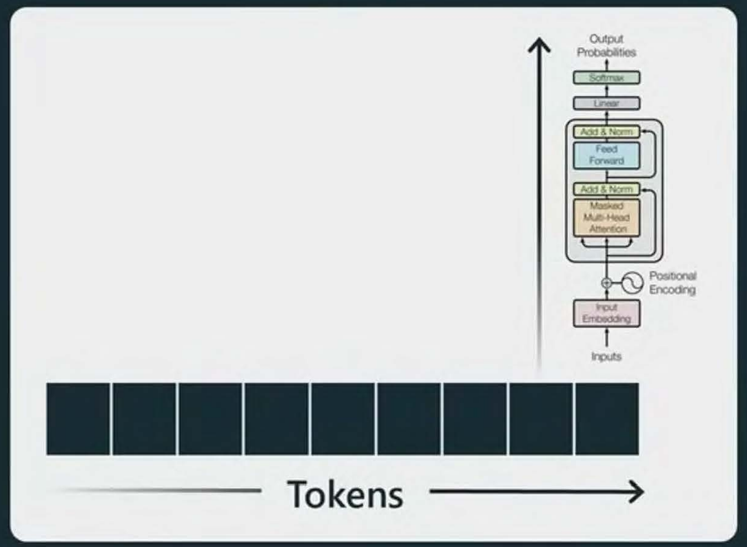
Human text generation vs. LLM text generation

"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)

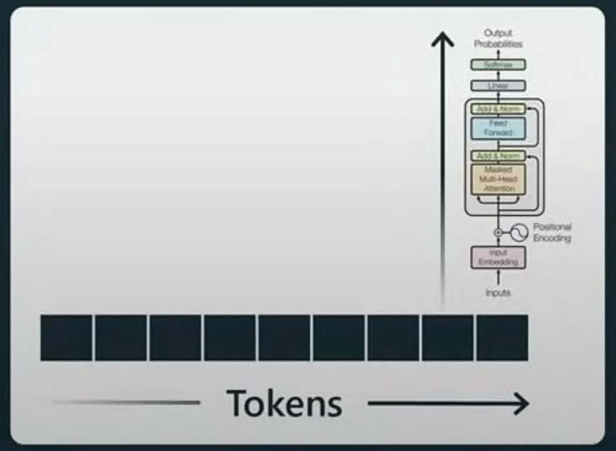
Chain of thought

"Models need tokens to think"

Break up tasks into multiple steps/stages

Prompt them to have internal monologue

Spread out reasoning over more tokens



(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)

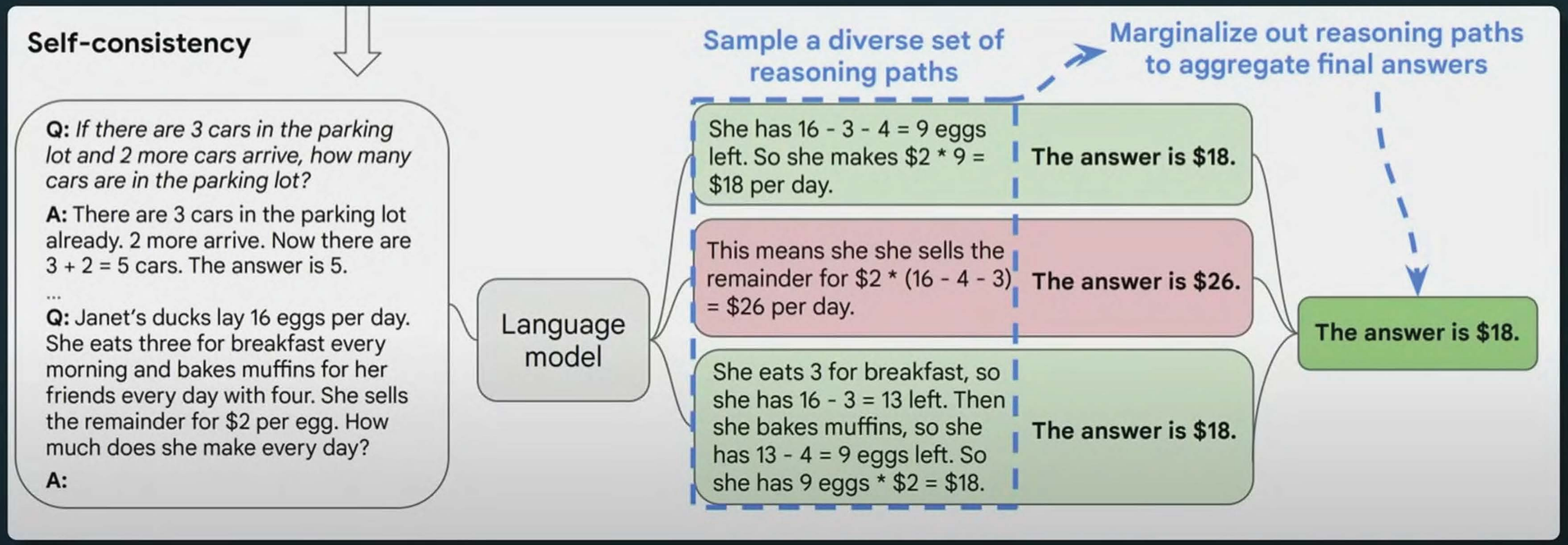
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(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.

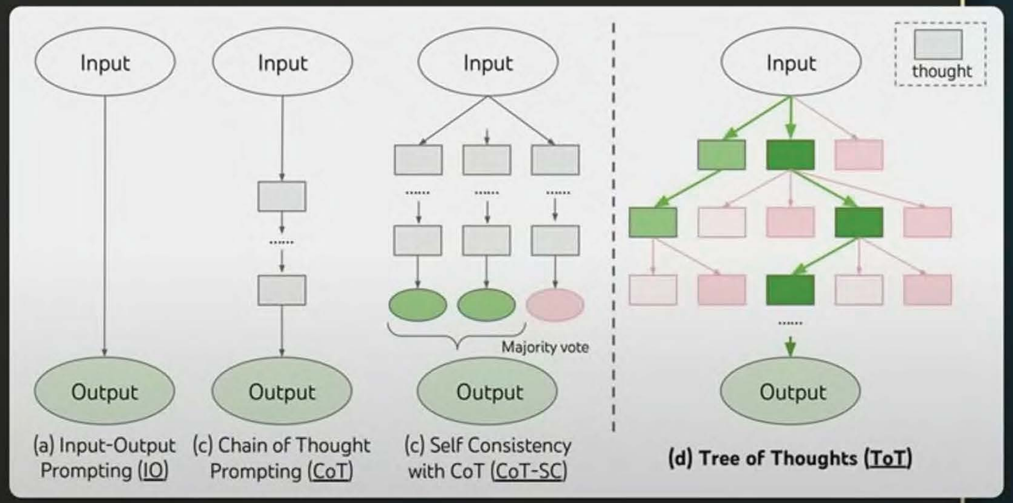
In the quietude, an ancient wisdom speaks,
Timeless stories of growth and decay,
A testament to resilience, perseverance seeks,
The rhythm of life, a dance, an eternal play.

[evjang.com/2023/03/26/self-reflection.html]

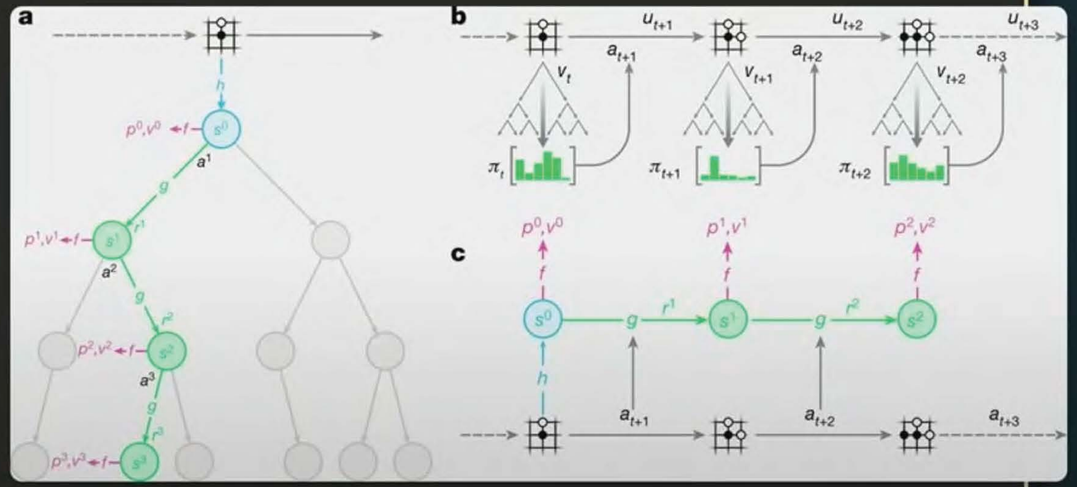
Recreate our 'System 2'

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:

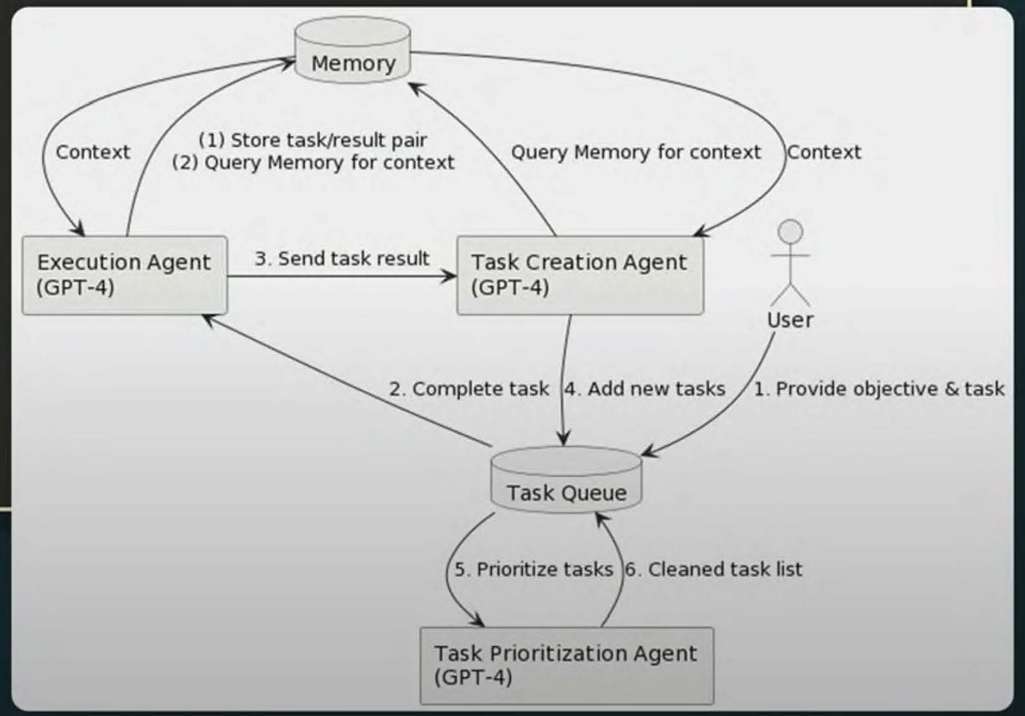


[Mastering the game of go without human knowledge, Silver at al. 2017 (AlphaGo)]
[Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao et al. 2013]

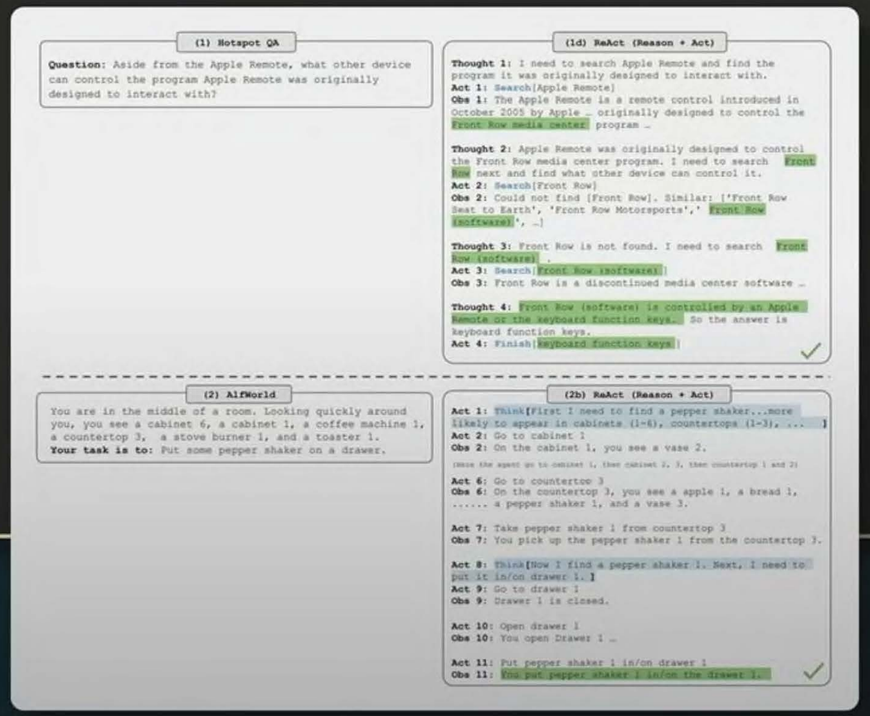
Chains / Agents

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”

...

[Large Language Models Are Human-Level Prompt Engineers, 2023]

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.

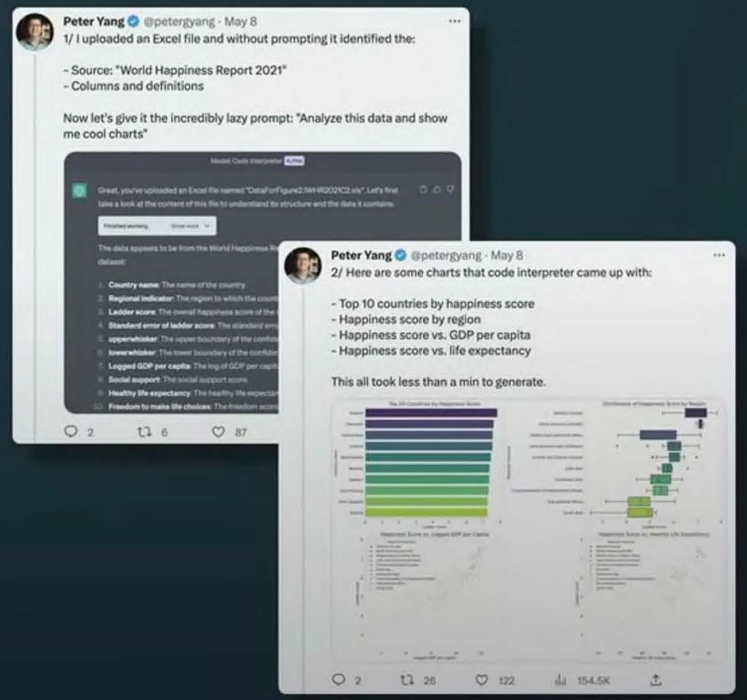
The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their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

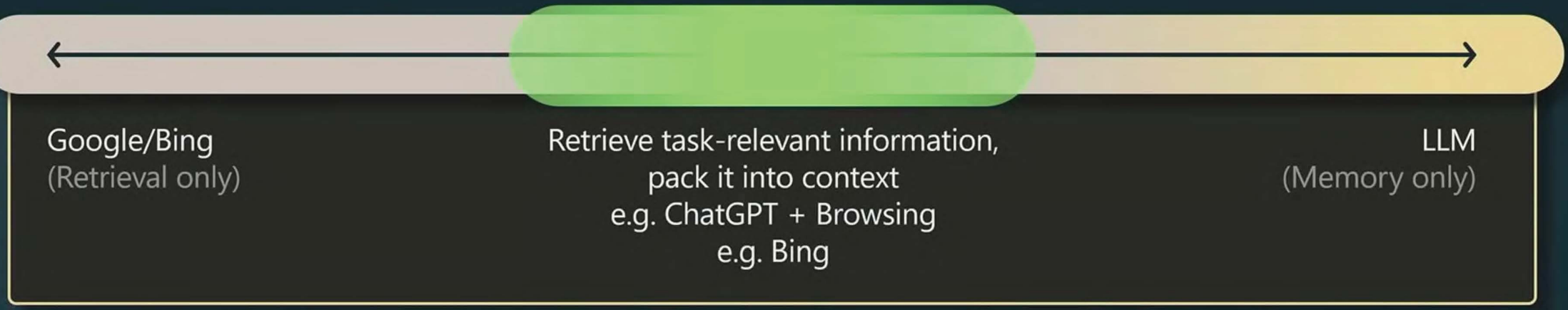
ChatGPT plugins

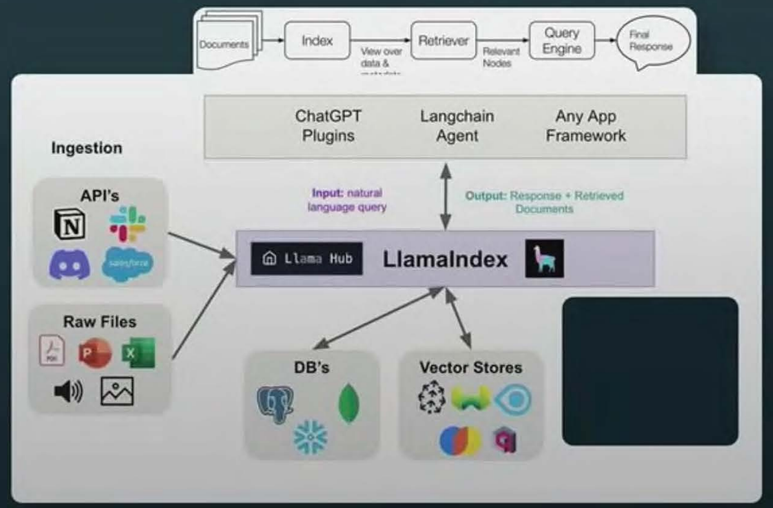
ChatGPT Code Interpreter



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

Constrained prompting

"Prompting languages" that interleave generation, prompting, logical control

{{guidance}}

```
# we use LLaMA here, but any GPT-style model will do
llama = guidance.llms.Transformers("your_path/llama-7b", device=0)

# we can pre-define valid option sets
valid_weapons = ["sword", "axe", "mace", "spear", "bow", "crossbow"]

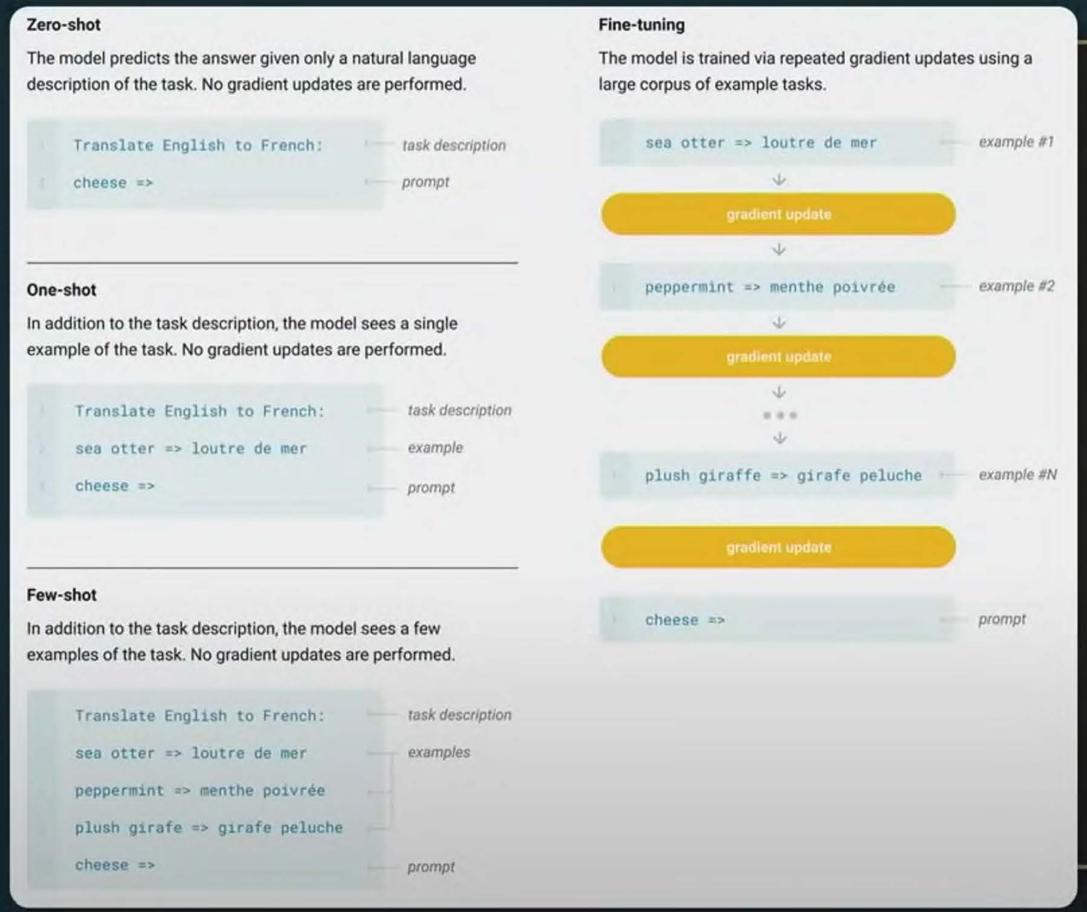
# define the prompt
character_maker = guidance("""The following is a character profile for an RPG game in JSON format.
```json
{
    "id": "{{id}}",
    "description": "{{description}}",
    "name": "{{gen 'name'}}",
    "age": {{gen 'age' pattern='[0-9]+' stop=','}},
    "armor": "{{#select 'armor'}}leather{{or}}chainmail{{or}}plate{{/select}}",
    "weapon": "{{select 'weapon' options=valid_weapons}}",
    "class": "{{gen 'class'}}",
    "mantra": "{{gen 'mantra' temperature=0.7}}",
    "strength": {{gen 'strength' pattern='[0-9]+' stop=','}},
    "items": [{{#geneach 'items' num_iterations=5 join=', '}}"{{gen 'this' temperature=0.7}}"{{/gen
}```""")

# generate a character
character_maker(
    id="e1f491f7-7ab8-4dac-8c20-c92b5e7d883d",
    description="A quick and nimble fighter.",
    valid_weapons=valid_weapons, llm=llama
)
```

The following is a character profile for an RPG game in JSON format.

```json
{
   "id": "e1f491f7-7ab8-4dac-8c20-c92b5e7d883d",
   "description": "A quick and nimble fighter.",
   "name": "Fighter",
   "age": 18,
   "armor": "plate",
   "weapon": "sword",
   "class": "fighter",
   "mantra": "I will protect the weak.",
   "strength": 10,
   "items": ["Hero's Hammer", "Fast-Healing Potion", "Magic Boots", "Shield of the Ancients", "Mystic Bow"]
}
```

github.com/microsoft/guidance/
```

Finetuning



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

[Language Models are Few-Shot Learners, Brown et al. 2020]

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 given 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,…

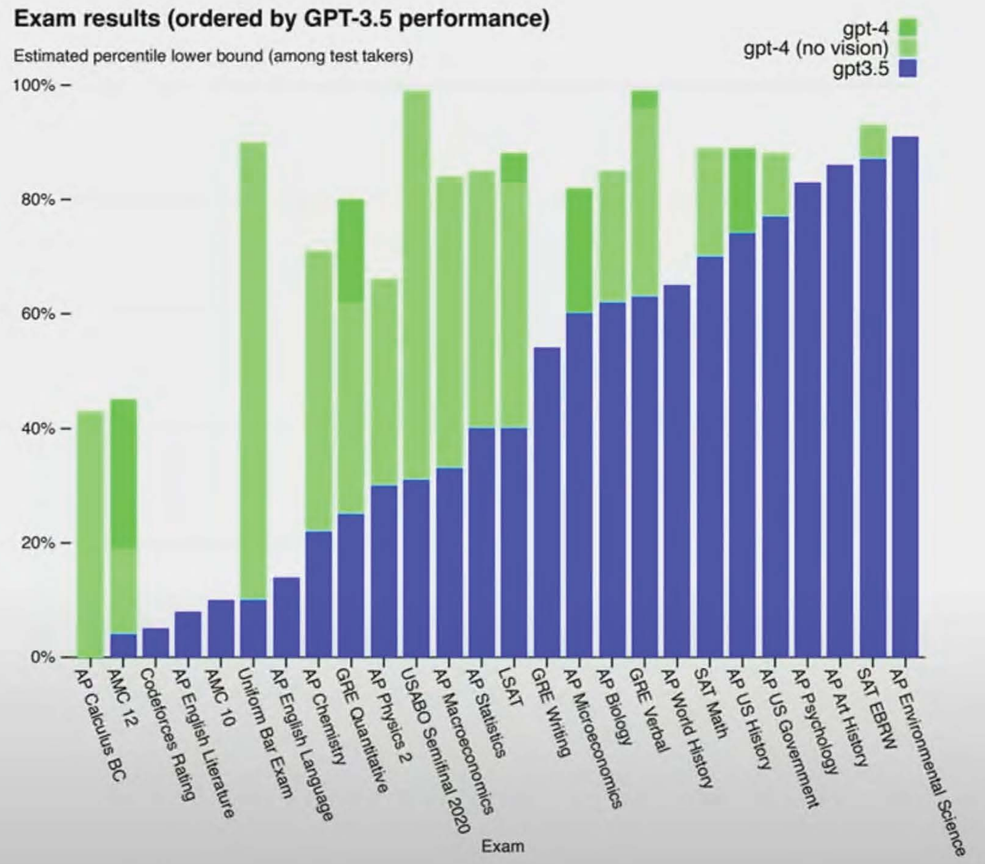
Recommendations:

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

→ Source of inspiration, suggestions

→ Copilots over autonomous agents

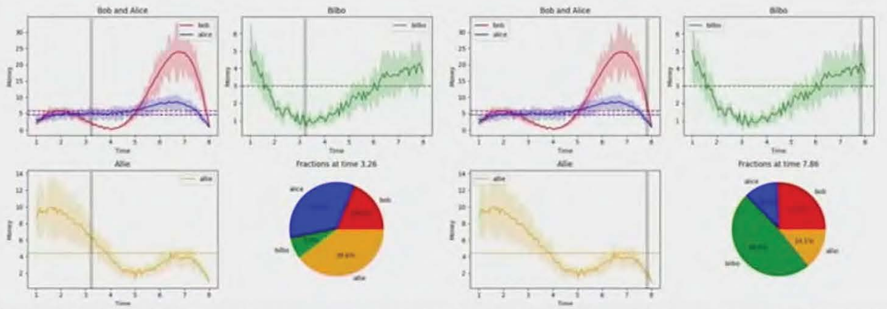
GPT-4



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]

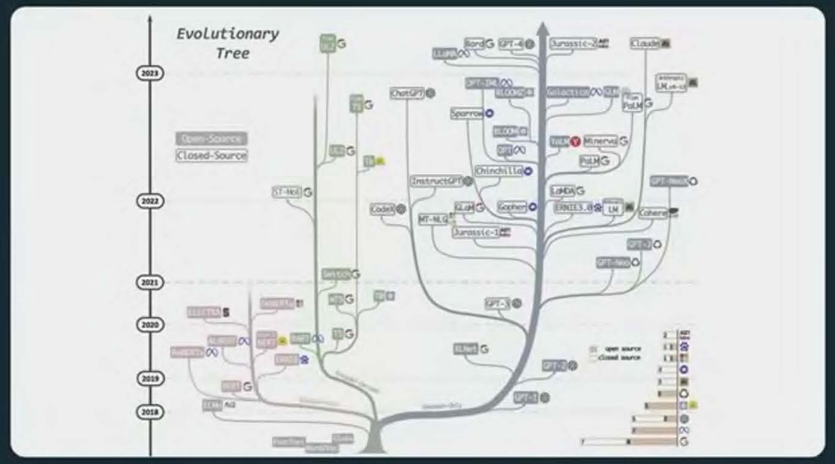


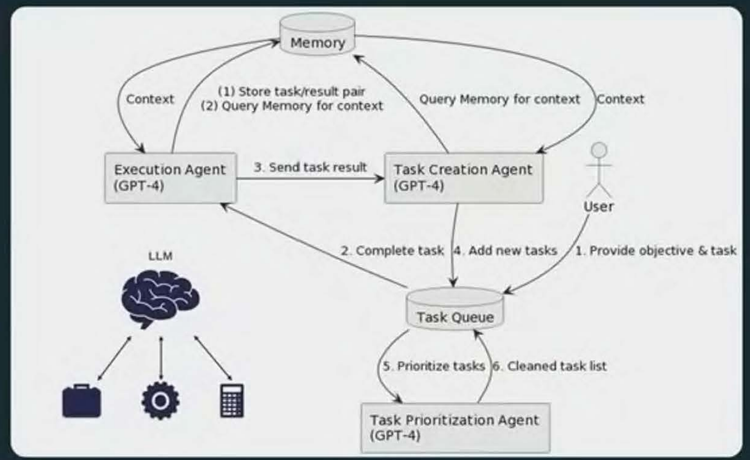
Prompt: Suppose $g(x) = f^{-1}(x), g(0) = 5, g(4) = 7, g(3) = 2, g(7) = 9, g(9) = 6$ what is $f(f(f(6)))$?

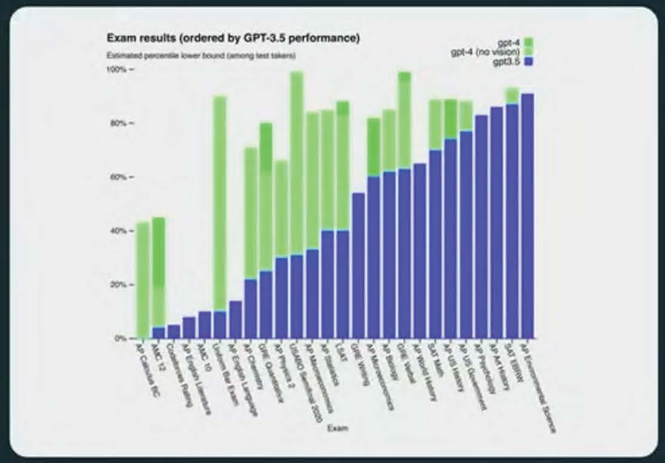
GPT-4: There are different ways to approach this problem, but one possible method is to use the fact that $g(x) = f^{-1}(x)$ implies that $f(g(x)) = x$ for any x in the domain of g. That is, applying f to both sides of the equation $g(x) = f^{-1}(x)$ gives $f(g(x)) = f(f^{-1}(x)) = x$. This means that 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, $f(f(f(6))) = f(f(9)) = f(7) = 4$.

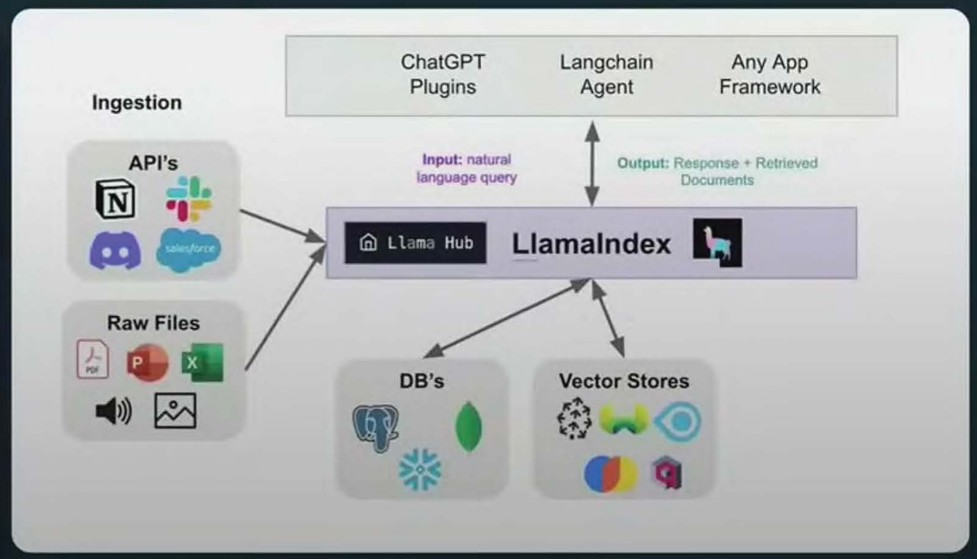
[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubuck et al. 2023]

Looking forward









The following is a character profile for an RPG game in JSON format.

```json
{
  "id": "e1f491f7-7ab8-4dac-8c20-c92b5e7d883d",
  "description": "A quick and nimble fighter.",
  "name": "Fighter",
  "age": 18,
  "armor": "plate",
  "weapon": "sword",
  "class": "fighter",
  "mantra": "I will protect the weak.",
  "strength": 10,
  "items": ["Hero's Hammer", "Fast-Healing Potion", "Magic Boots", "Shield of the Ancients", "Mystic Bow"]
}
```

```python
import openai

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

```python
import openai

response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Can you say something to inspire the audience of Microsoft BUILD 2023?"},
    ]
)
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!