

Understanding Large Language Models

Carsten Eickhoff, Michael Franke and Polina Tsvilodub

Session 03: State-of-the-art LLMs, in-context learning & prompt-engineering

Main learning goals

1. taking stock

- recap RNN, LSTM, Transformer

2. Large Language Models (2020-2022)

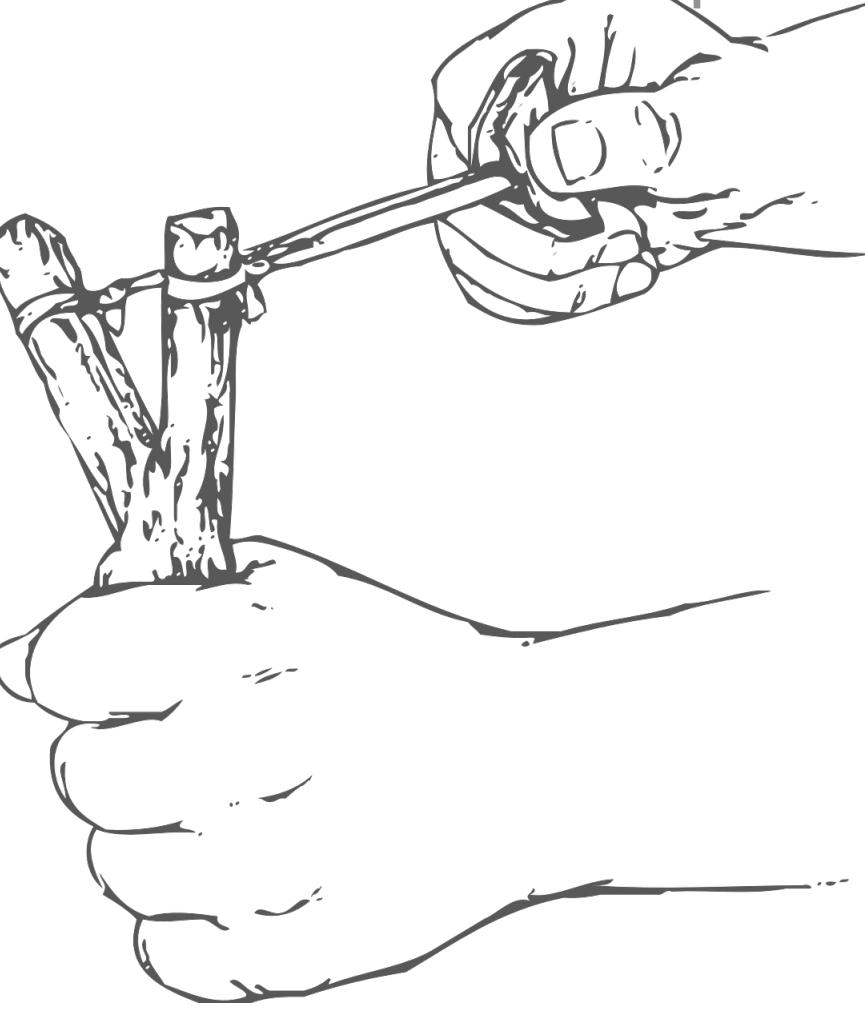
- close-up LLaMA

3. in-context learning

- few-shot learning
- prompt understanding

4. sophisticated prompting

- **c**hain-of-thought
- self-consistency
- knowledge generation
- tree-of-thought





**Taking stock &
looking ahead**

Language & sequence models

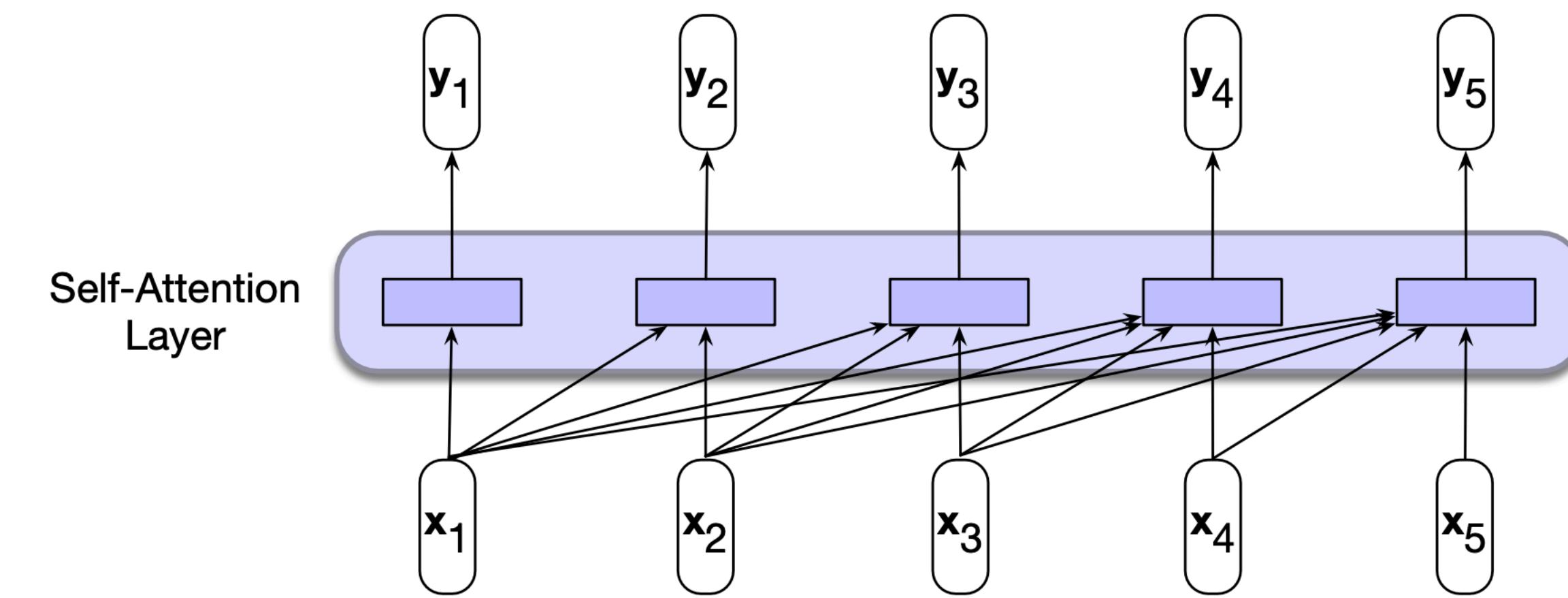
welcome to the (terminological) jungle

- ▶ **language model** (wide definition)
 - any kind of model used to perform tasks with natural language
- ▶ **language model** (narrow definition)
 - probabilistic model of natural language
- ▶ **neural language model** (super narrow definition)
 - neural network predicting natural language, given some input
 - autoregressive, masked, image captioning ... whatever
- ▶ **embedding**
 - vector representation of a chunk of natural language
 - used for many different tasks, including language modeling and more
 - can arise as a 'by-product' of language modeling
- ▶ **sequence model**
 - model that processes and/or generates a token sequence

Autoregressive / left-to-right

RNN

LSTM

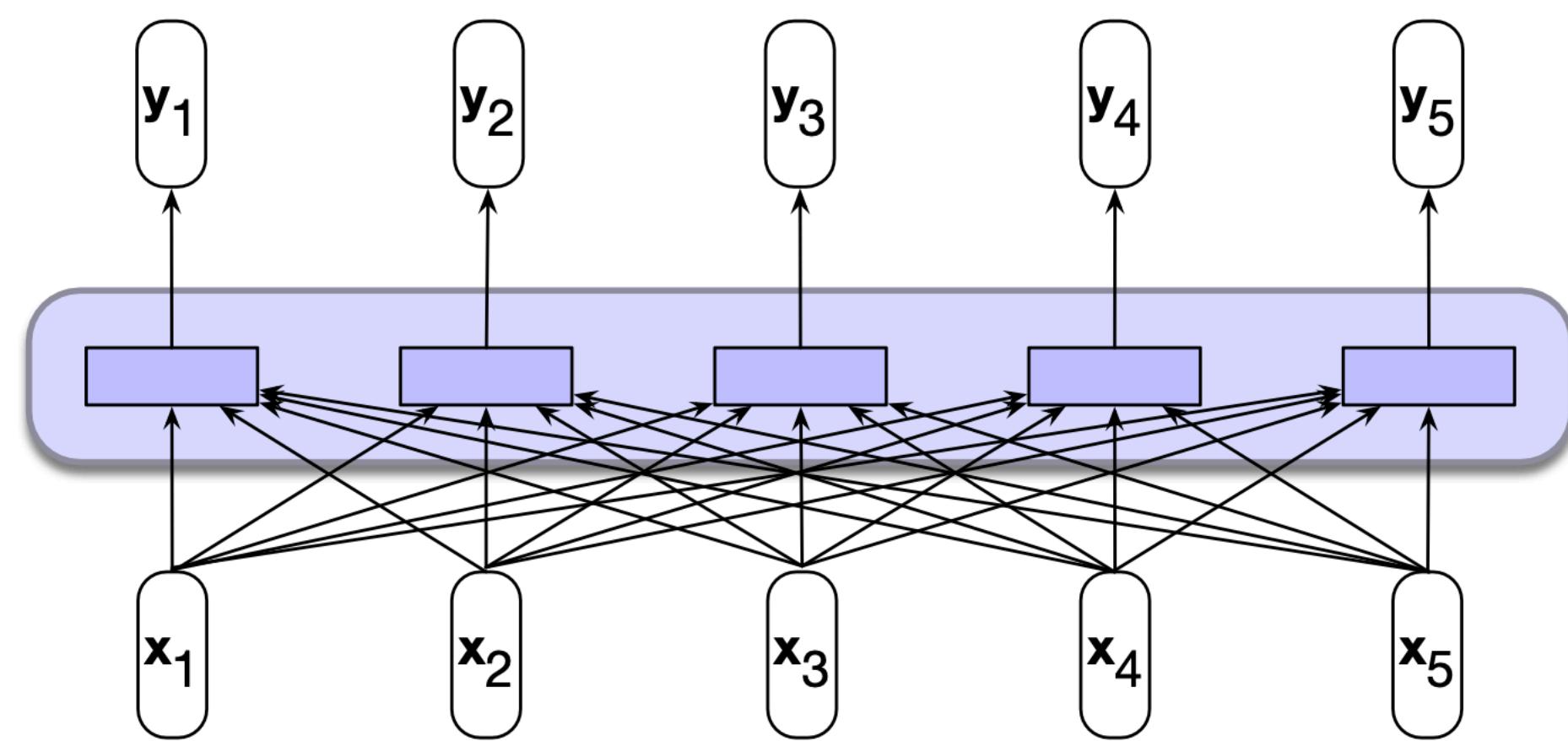


computation for input $\mathbf{x}_1, \dots, \mathbf{x}_3$ **blind** to \mathbf{x}_4 and \mathbf{x}_5

\mathbf{y}_5 is embedding for input $\mathbf{x}_1, \dots, \mathbf{x}_5$
 \mathbf{y}_5 is a “left-contextual embedding”

Bidirectional

Self-Attention Layer



computation for input $\mathbf{x}_1, \dots, \mathbf{x}_3$ **sees** \mathbf{x}_4 and \mathbf{x}_5

$\mathbf{y}_1, \dots, \mathbf{y}_5$ is embedding for input $\mathbf{x}_1, \dots, \mathbf{x}_5$
 \mathbf{y}_i are bidirectional “contextual embeddings”

Different kinds of sequence processing models

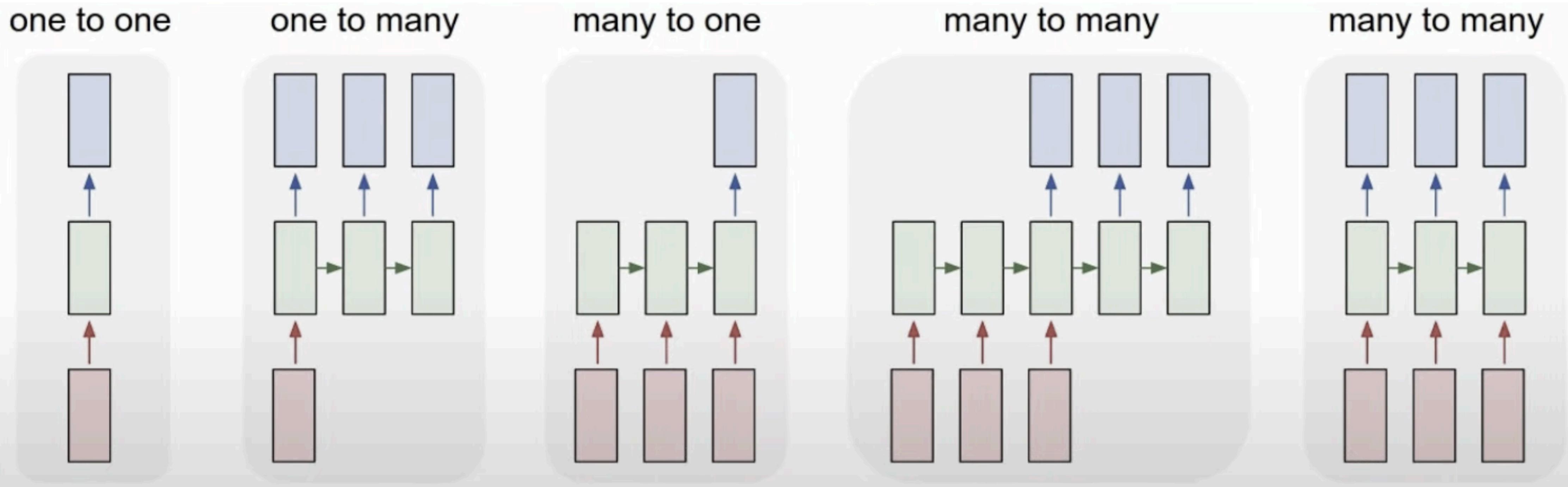
sequence as input and/or (simultaneous) output

One to One: Classification.

One to Many: Image Captioning.

Many to One: Sentiment Analysis.

Many to Many: Machine Translation.



Think break:

Which of these architectures go with which kinds of applications?

Where, if at all, would you locate autoregressive / masked language modeling? **Many to many**

Memory, attention & relevance

in humans & machines

- ▶ **human memory**
 - optimized to retrieve context-relevant info when cued
 - optimized to store information deemed relevant
- ▶ **human attention**
 - filters external and internal stimuli based on relevance

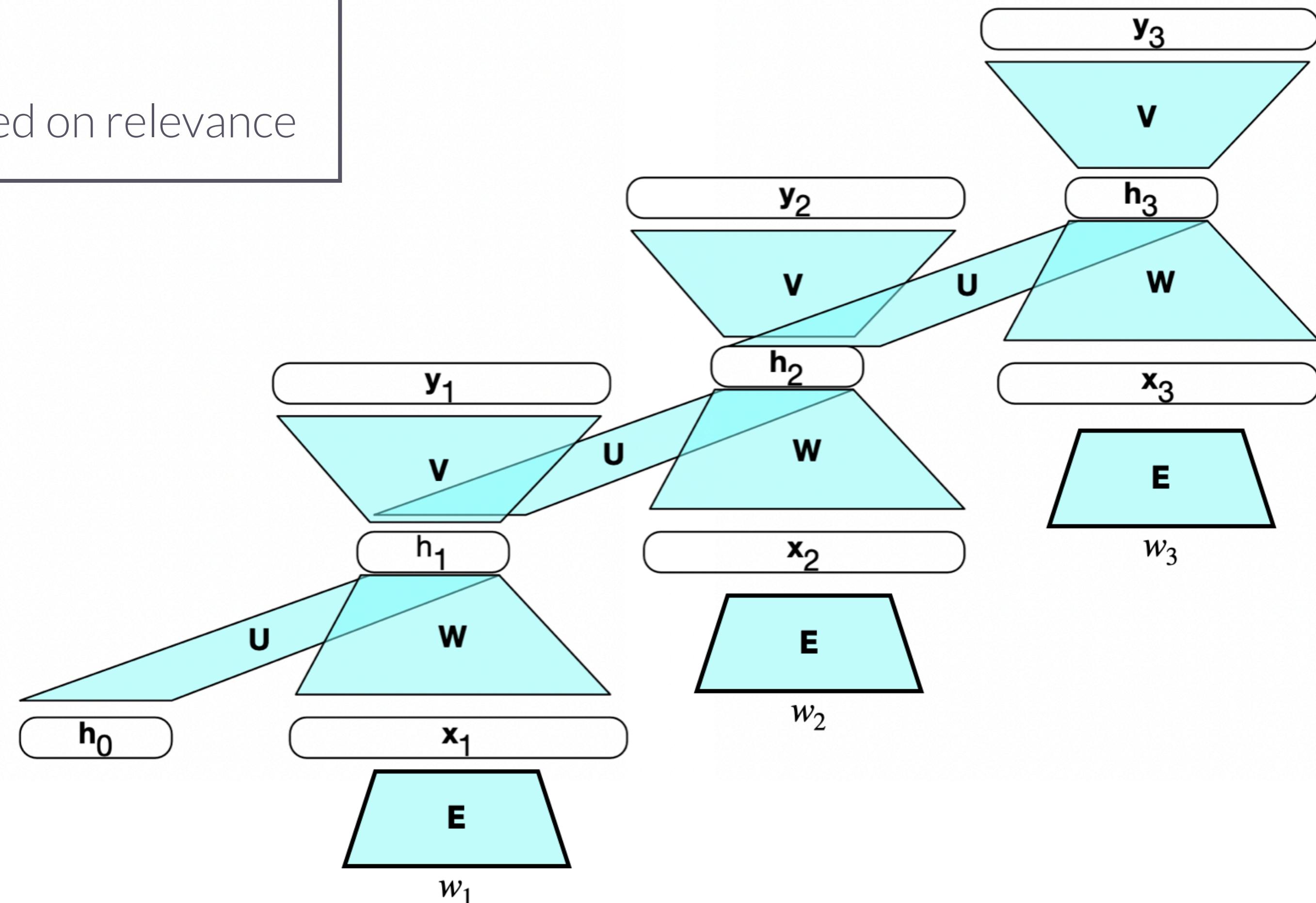
Memory, attention & relevance

in humans & machines

- ▶ **human memory**
 - optimized to retrieve context-relevant info when cued
 - optimized to store information deemed relevant

- ▶ **human attention**
 - filters external and internal stimuli based on relevance

RNNs

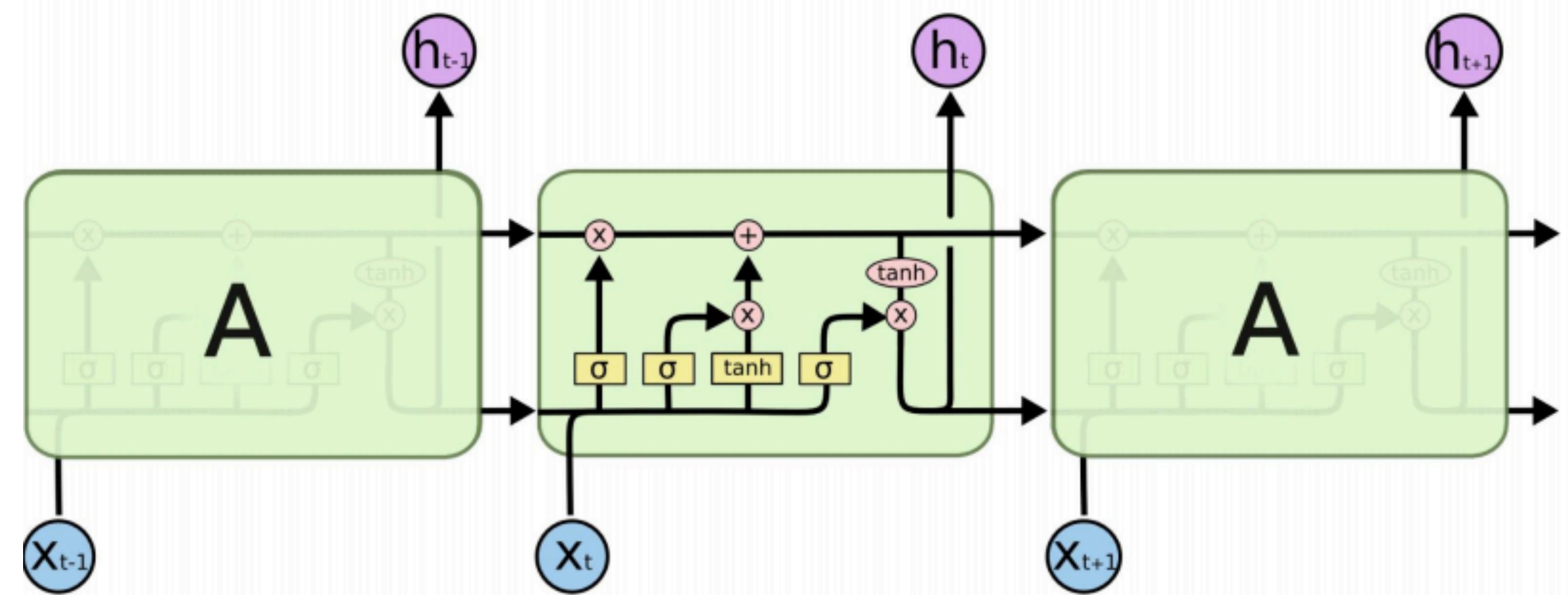


Memory, attention & relevance

in humans & machines

- ▶ human memory
 - optimized to retrieve context-relevant info when cued
 - optimized to store information deemed relevant
- ▶ human attention
 - filters external and internal stimuli based on relevance

LSTMs



Memory, attention & relevance

in humans & machines

- ▶ **human memory**
 - optimized to retrieve context-relevant info when cued
 - optimized to store information deemed relevant

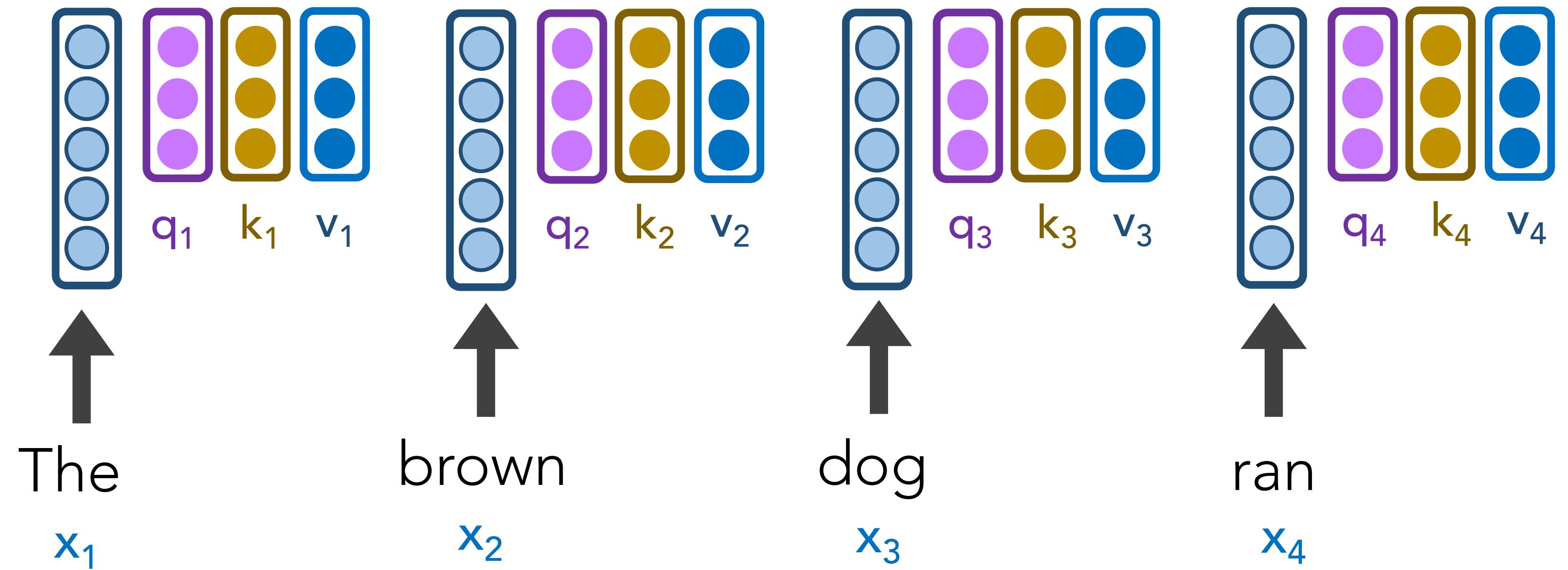
- ▶ **human attention**
 - filters external and internal stimuli based on relevance

query: what's relevant for me now?

key: what kind of info do I have?

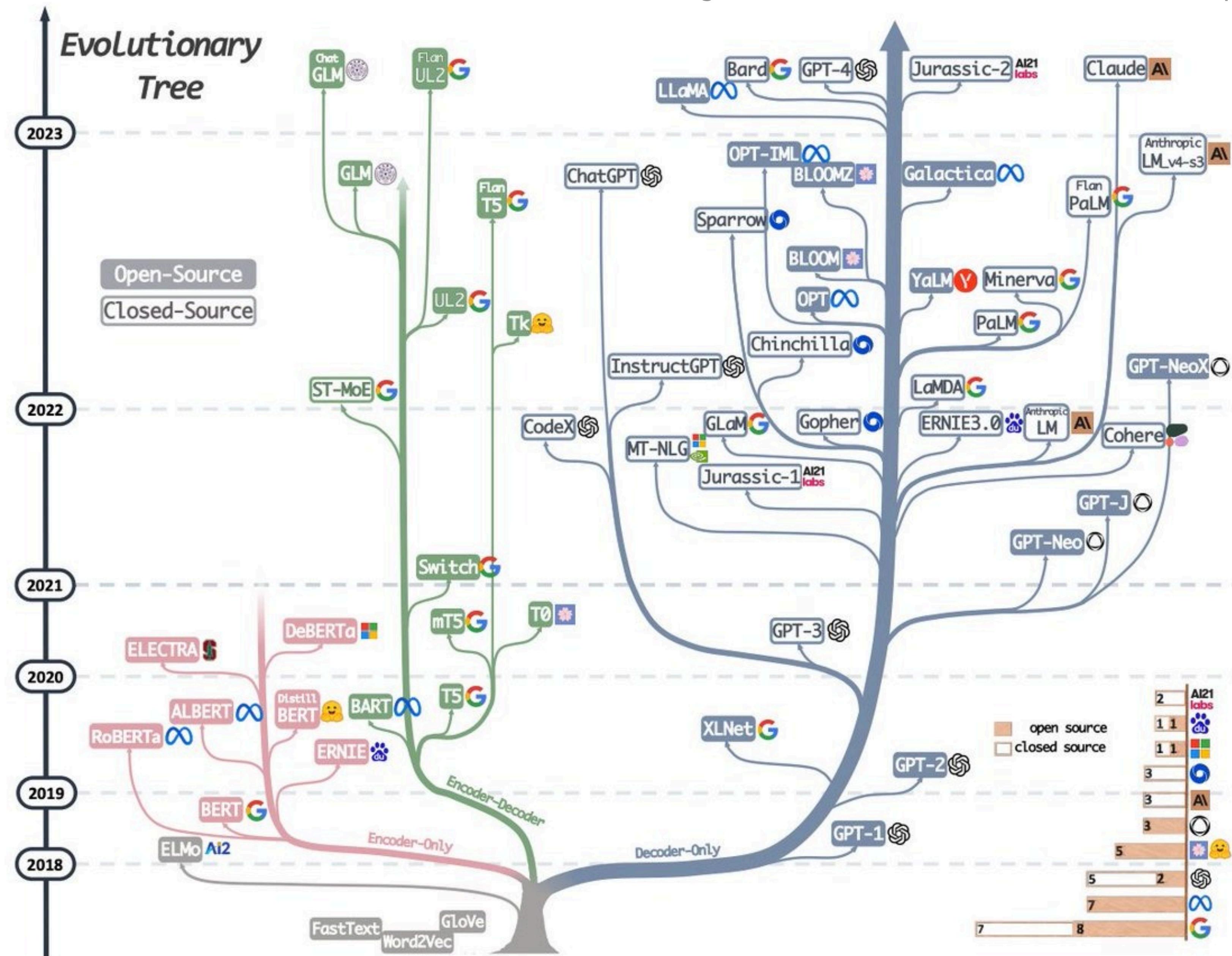
value: what concrete info do I have?

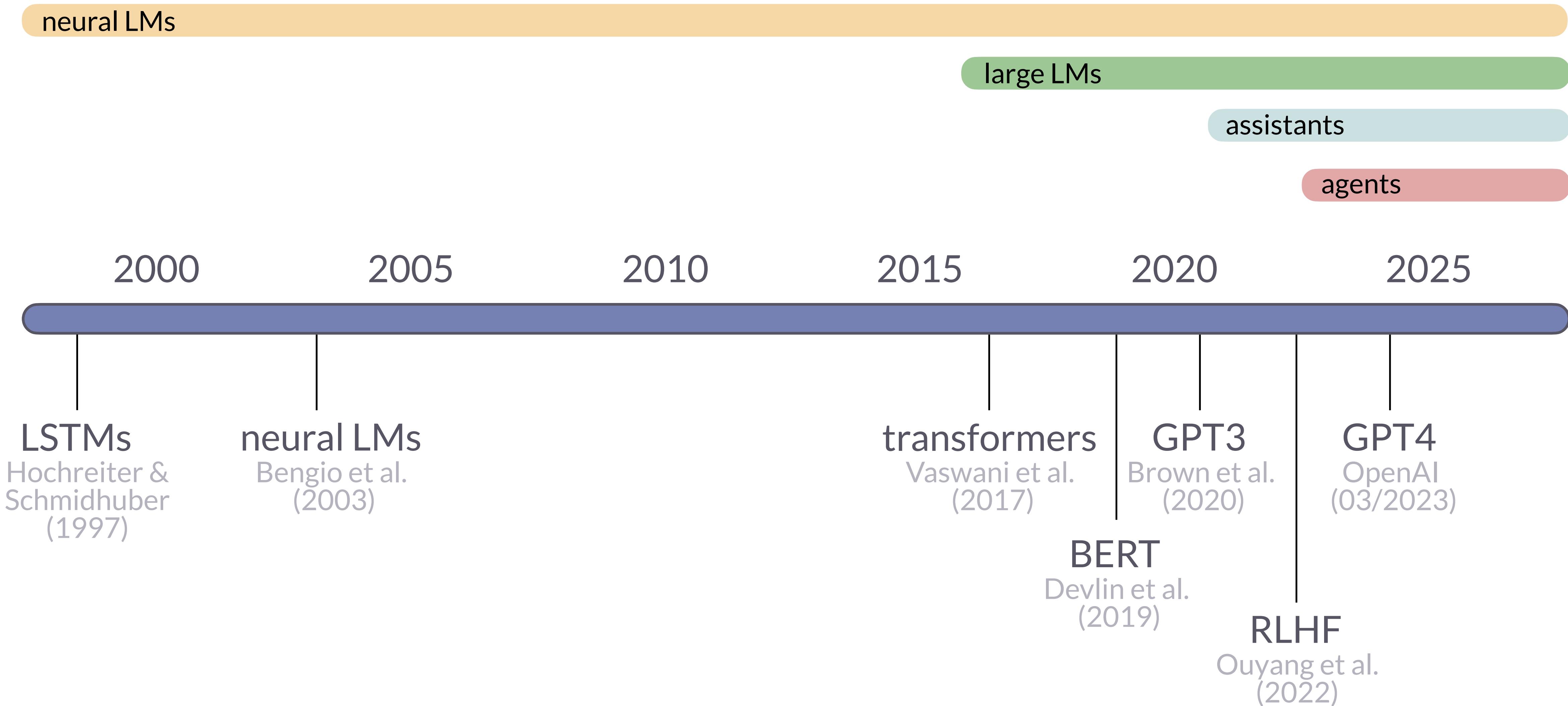
Transformers





State-of-the-art LLMs





Kinds of Large Language Models

core LLMs

(foundation models)

- ▶ predict statistically likely next token
- ▶ e.g.,
 - GPT-2
 - LLaMA2 / LLaMA3
 - ...

today

prepped LLMs

(assistants)

- ▶ fine-tuned (e.g. RLHF)
- ▶ predict token likely to please the user
- ▶ e.g.,
 - GPT-3.5
 - LLaMA3 Instruct
 - ...

next session

LLM-based applications

(agents)

- ▶ algorithm using LLMs
- ▶ e.g.:
 - sophisticated prompting
 - Chat-GPT w/ tools
 - AutoGPT
 - neuro-symbolic models
 - ...

next to next session

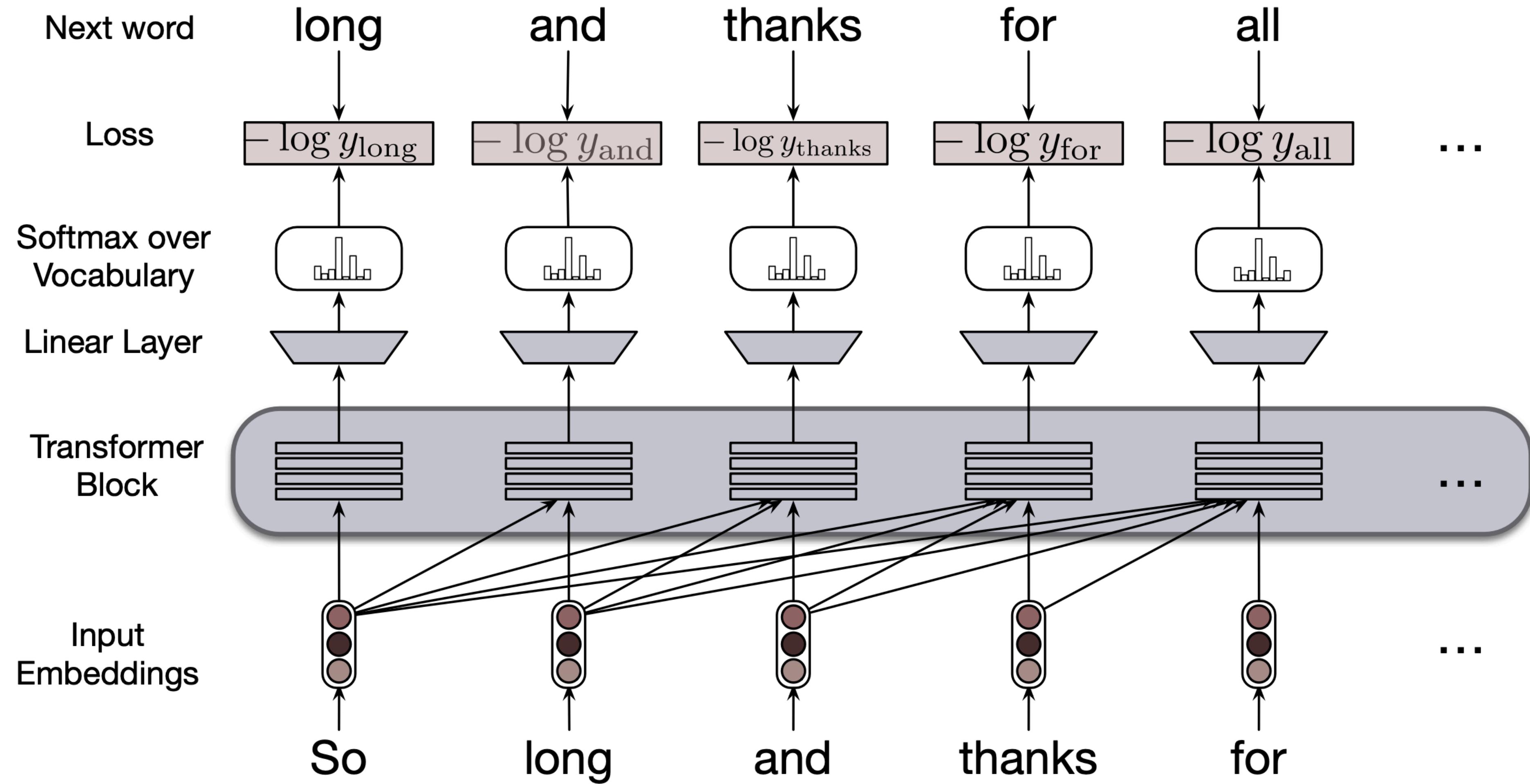
WEIRD WYOMING

- ▶ just as experimental psychology is **WEIRD**
 - **Western**
 - **Educated**
 - **Industrialized**
 - **Rich**
 - **Democratic**
- ▶ usual LLM training data is from **WYOMING**
 - **Western**
 - **Young**
 - **Opinionated**
 - **Males with**
 - **Internet from**
 - **Non-marginalized**
 - **Groups**



Language modeling objective

maximize token-in-context probability



The LLaMA model family

Large Language Model Meta AI

Name	Release date	Parameters	Context length	Corpus size	Commercial viability?
LLaMA	February 24, 2023	<ul style="list-style-type: none"> • 6.7B • 13.0B • 32.5B • 65.2B 	2048	1.0-1.4T	No
Llama 2	July 18, 2023	<ul style="list-style-type: none"> • 6.7B • 13.0B • ~34B (unreleased) • 69B 	4096	2T	Yes
Llama 3	April 18, 2024	<ul style="list-style-type: none"> • 8B • 70.6B • 400B+ (unreleased) 	8912	15T	Yes

base model only

base model,
fine-tuned & chat

base & instruction

<https://en.wikipedia.org/wiki/LLaMA>

research reports:

LLaMA ⇒ Touvron, Lavril et al. (2023)

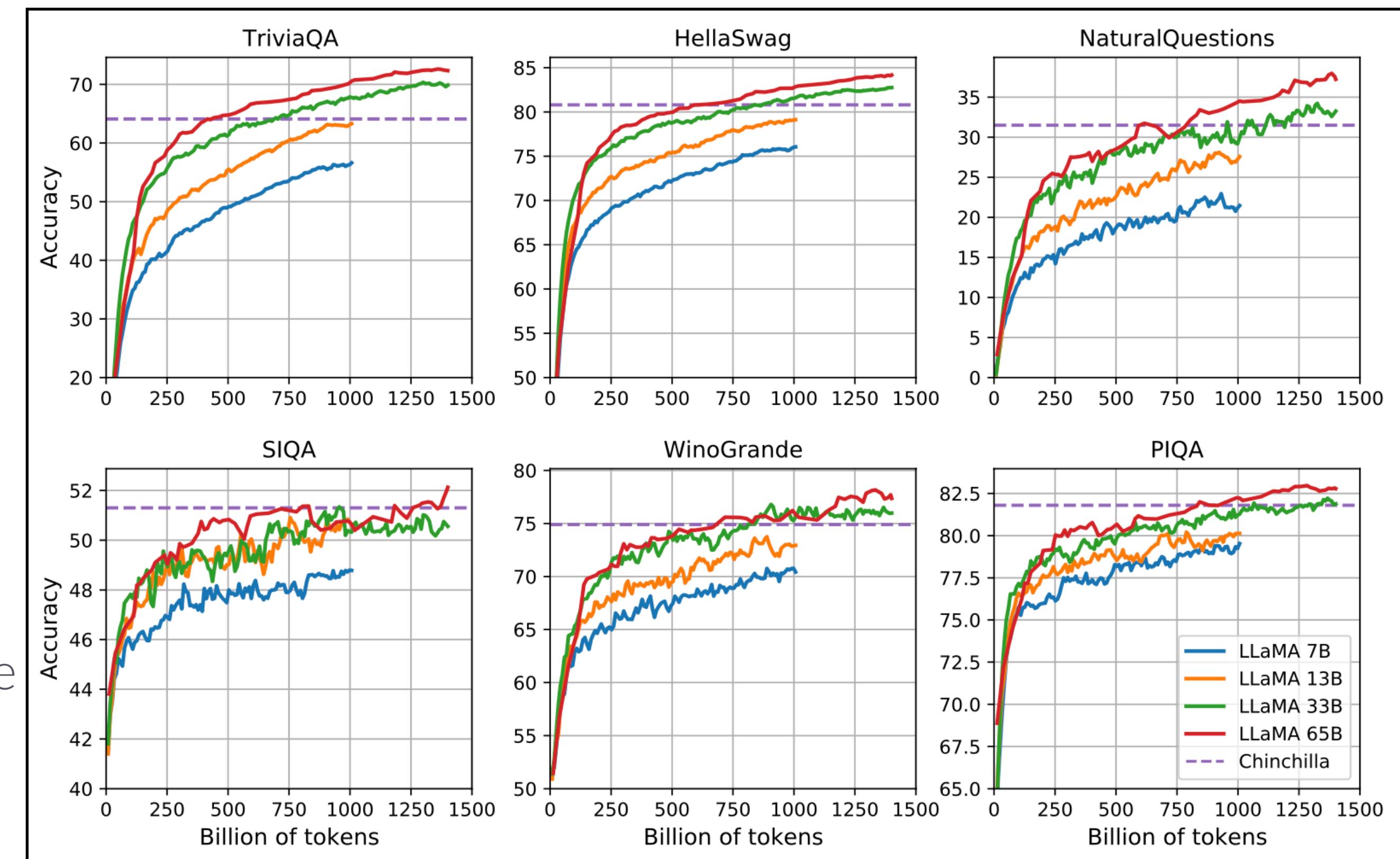
Llama-2 ⇒ Touvron, Martin et al. (2023)

Llama-3 ⇒ ... pending ...

Building LLaMA

motivation

- ▶ trained on publicly available data only
- ▶ weights released on request
 - restricted commercial use
- ▶ two ways of spending compute
 - (1) larger model w/ same training volume
 - (2) smaller model w/ more training [← LLaMA]
 - (2) is cheaper at inference time
- ▶ 7B, 13B, 70B parameter models
 - 13B LLaMA matches best GPT-3 model of the time
 - the latter at 175B parameters
 - 70B LLaMA matches leading models of the time Chinchilla (70B) and PaLM (540B)



Building LLaMA

training details

- ▶ **training data set:**
 - various sources, all preprocessed, see →
- ▶ **tokenization**
 - byte-pair encoding \Rightarrow 1.4T tokens (!)
- ▶ **model architecture**
 - autoregressive transformer w/ tweaks
 - normalize input, not output of transformer sublayers
 - replace ReLU activation w/ SwiGLU (Shazeer 2020)
 - rotary positional embeddings
- ▶ **optimizer**
 - AdamW w/ carefully chosen parameters
 - the latter at 175B parameters
 - 70B LLaMA matches leading models of the time
Chinchilla (70B) and PaLM (540B)
- ▶ **compute optimization** (see paper)

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.

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

Building LLaMA

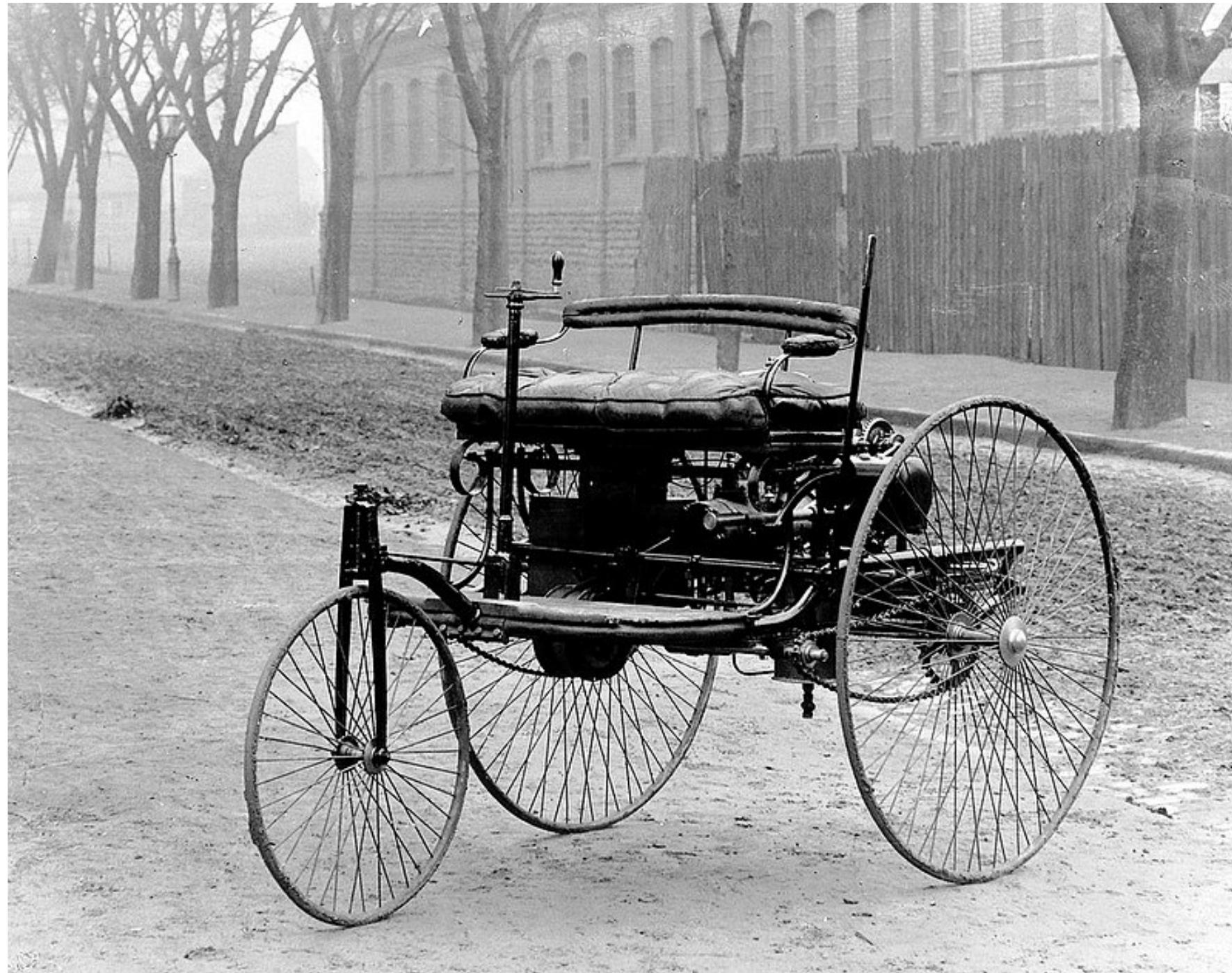
example results

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	88.0	82.3	-	83.4	81.1	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	80.0	57.8	58.6
	65B	85.3	82.8	52.3	84.2	77.0	78.9	56.0	60.2

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

On engineering, or how to overpower a simple idea

you've come a long way, baby



Benz Patent-Motorwagen (1885)



Red Bull Racing RB18 (2022)

In-Context Learning

LLMs respond to their context,
learn from the context,
change their behaviours and response to their context

demo



Zero-shot prompting

- ▶ may include task instruction, but has no example or illustration
 - works well only for models fine-tuned on instruction-following data
 - works well only for frequent (simple) tasks

INPUT

Classify the sentence as positive, neutral or negative.
Sentence: This class is super exciting!
Sentiment:

OUTPUT

positive

In-context learning

a.k.a. k-shot learning

- ▶ prompt with k pairs of **demonstrations** (x_i, y_i)
 - x_i is the **input**
 - y_i is the **output** or **label**
- ▶ add only the target input x_t of test item (x_t, y_t)
- ▶ ICL is an emerging ability
 - boosts performance on common tasks, even though models are NOT trained do in-context learning
 - shown first for GPT-3 by [Brown et al. \(2020\)](#)
- ▶ works with or without (!) task instructions
- ▶ common observation:
 - performance increases with size k of demonstrations

INPUT

This class is interesting. neutral
This class is my favorite. positive
This class is a drag. negative.
This class is super exciting.

OUTPUT

positive

In-context learning

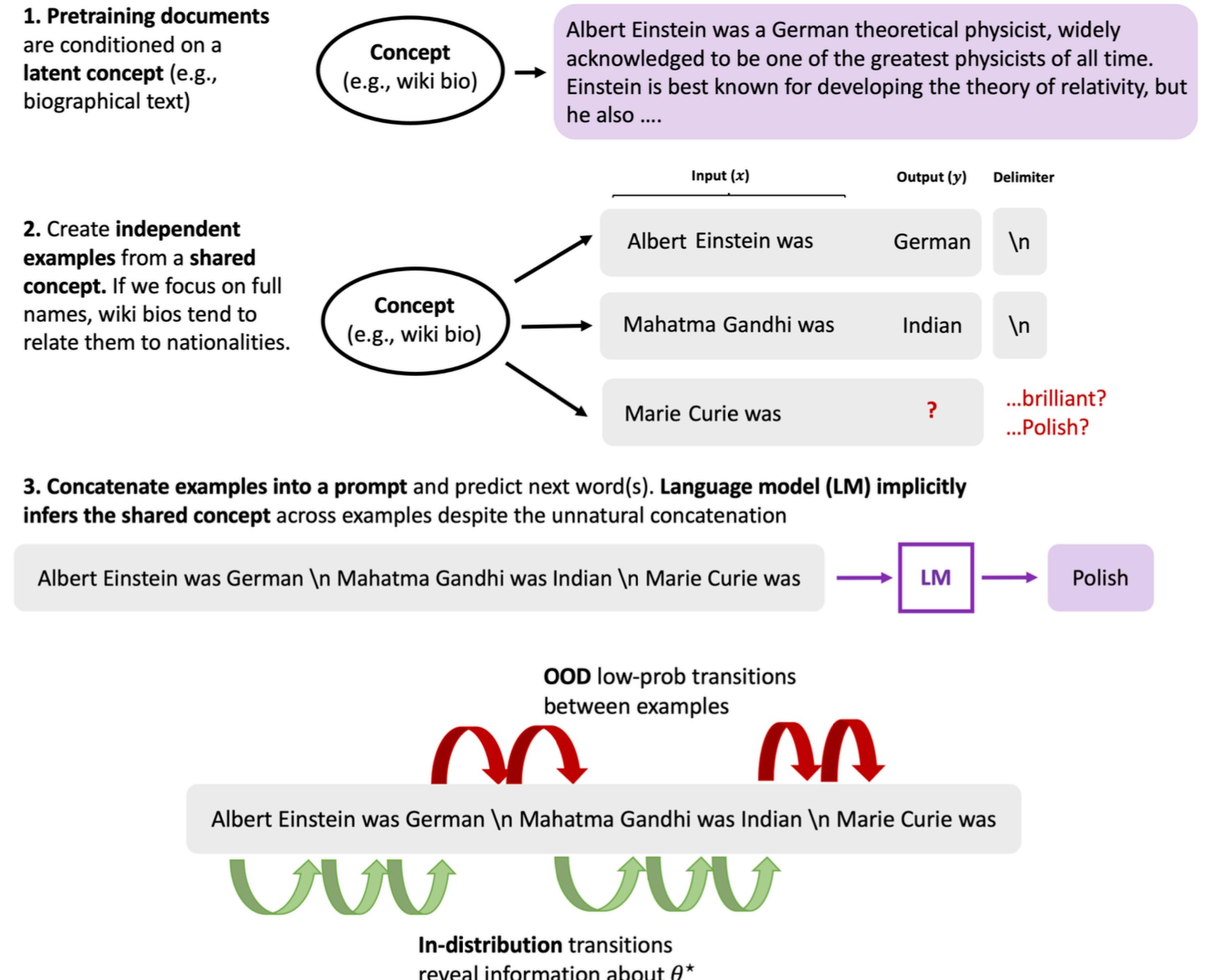
to be covered next

- ▶ ICL as implicit Bayesian inference
 - Xie et al. (2022)
- ▶ ICL may not be “learning” proper
 - Reynolds & McDonell (2021)
- ▶ conditions influencing ICL performance
 - Min et al. (2022)
- ▶ ICL without “prompt understanding”
 - Webson & Pavlick (2022)
- ▶ ICL improves with additional explanations
 - Lampinen et al. (2022)

In-context learning as implicit Bayesian inference

Xie et al. (2022)

- ▶ **starting point:**
 - training data is generated by a **hierarchical stochastic process**
 - this is a more commonly expressed idea
- ▶ **observation:**
 - k-shot input is out-of-distribution
 - high within-example probability
 - low between-example probability
- ▶ **conjecture:**
 - in-context learning is inference of the **latent type** shared by demonstrations

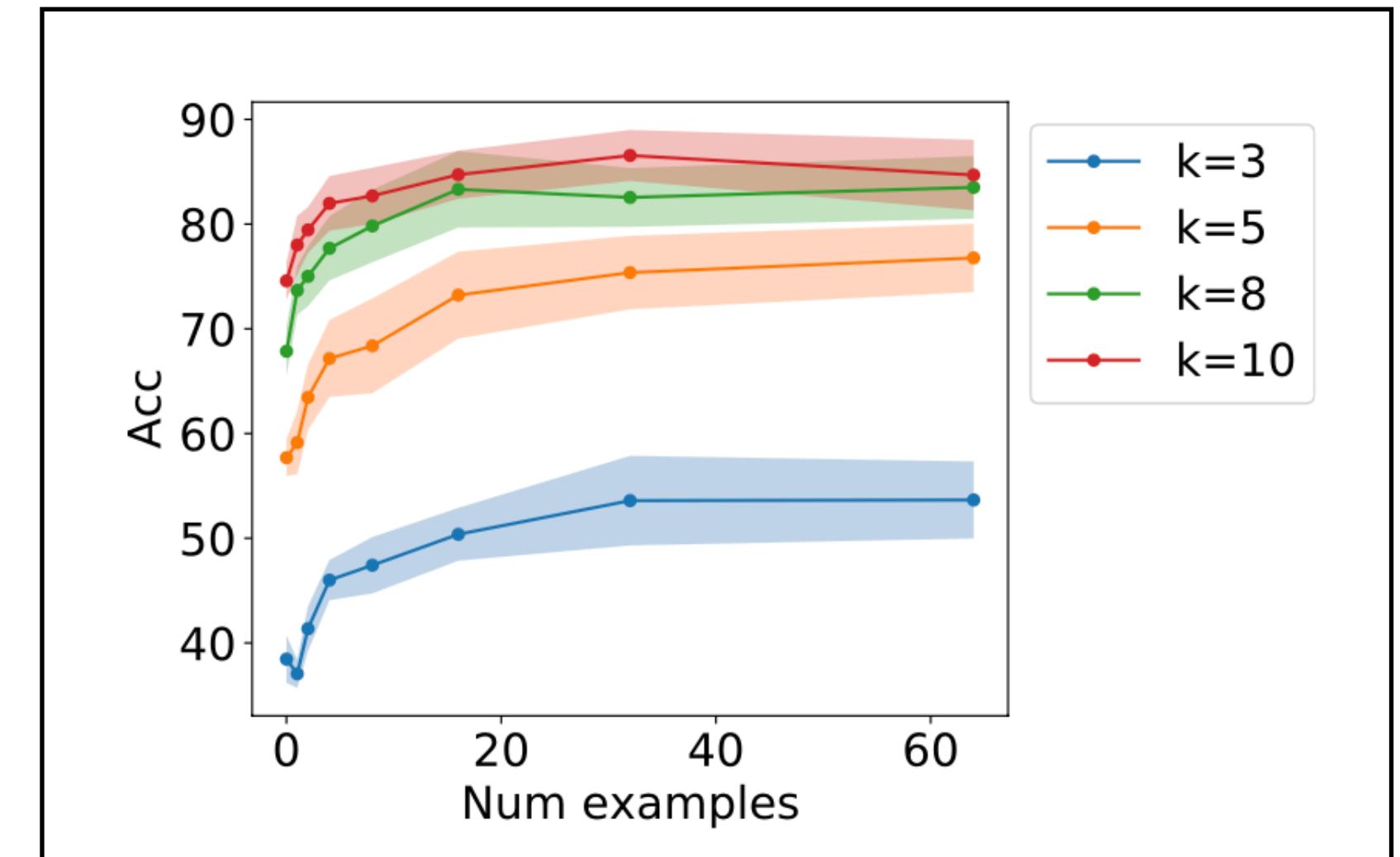


In-context learning as implicit Bayesian inference

Xie et al. (2022)

- ▶ theoretical argument:
 - proof that recovering the latent document type from k -shot examples is theoretically possible
- ▶ simulation results:
 - generate synthetic training data from a **hierarchical HMM process**
 - train autoregressive LSTM and transformer models (small-ish)
- ▶ results:
 - both LSTMs and transformers show **emerging in-context learning ability**
 - reproduces known effects:
 - more examples \Rightarrow better performance
 - example ordering matters
 - larger models \Rightarrow better in-context learning performance at equal predictive accuracy on train/test data

effect of demonstration size



Towards “prompt engineering”

Reynolds & McDonell (2021)

- ▶ contra claim “k-shot learning = meta-learning” sensu strict
- ▶ result: effect of examples can be retrieved by proper prompt
- ▶ interpretation: task ability comes from pre-training; prompt guides the system towards the “task location”

simple colon prompt

```
French: example_source_phrase
English: example_target_phrase

French: example_source_phrase
English: example_target_phrase

[...]

French: source_phrase
English:
```

master translator prompt

```
A French phrase is provided: source_phrase
The masterful French translator flawlessly
translates the phrase into English:
```

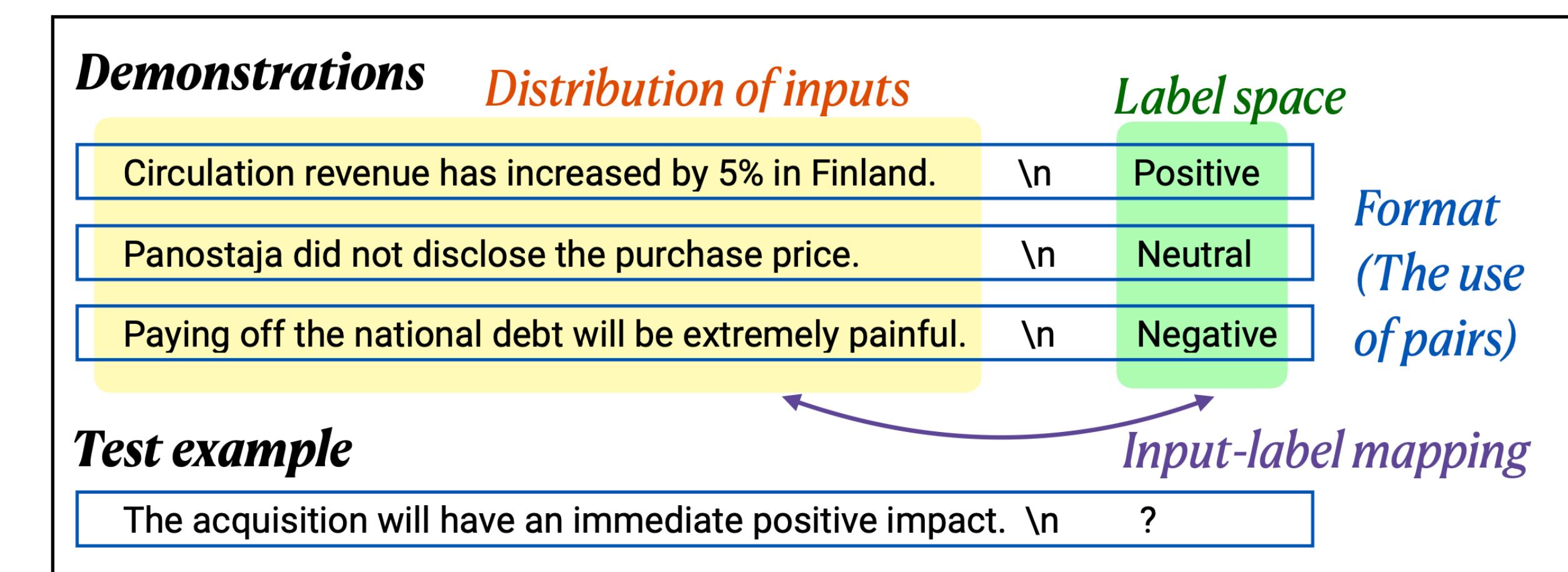
results

Prompt	Babbage / 6.7B	Curie / 13B
OpenAI 0-shot	15.5	22.4
OpenAI 1-shot	31.6	31.4
OpenAI 64-shot	36.4	38.3
Reproduced OpenAI 0-shot	15.9	18.7
Reproduced OpenAI 1-shot	21.8	24.1
Reproduced OpenAI 10-shot	25.1	27.9
Simple colon 0-shot	23.5	33.3
Simple colon 1-shot	18.0	27.6
Simple colon 10-shot	24.1	33.4
Master translator 0-shot	26.5	32.9

What makes in-context learning work?

Min et al. (2022)

- ▶ empirically test effect of four factors:
 - **correctness** of input-label mapping
 - **similarity** of examples and target input
 - **coverage** of label space in examples
 - **format** of input-label examples
- ▶ main results:
 - input-label correctness matters little
 - other factors matter more

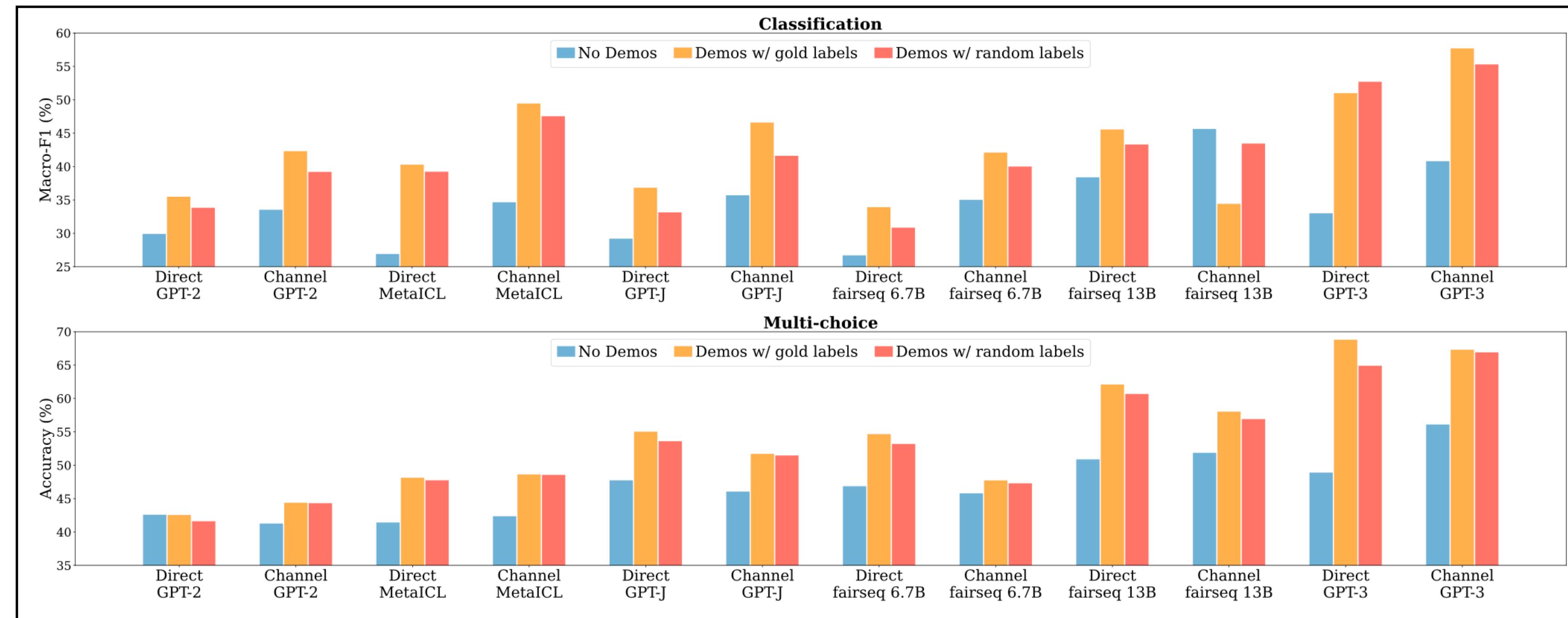


What makes in-context learning work?

Min et al. (2022)

► methods:

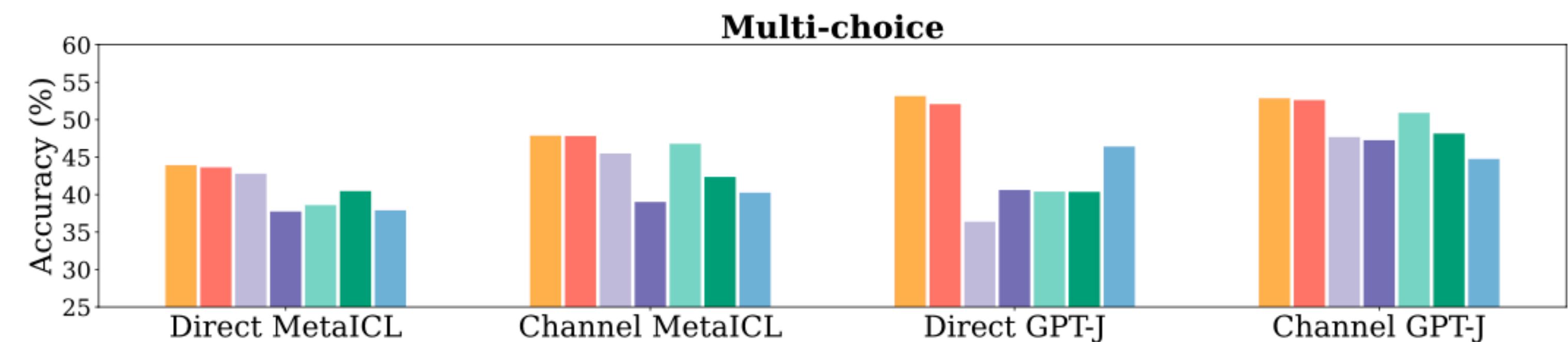
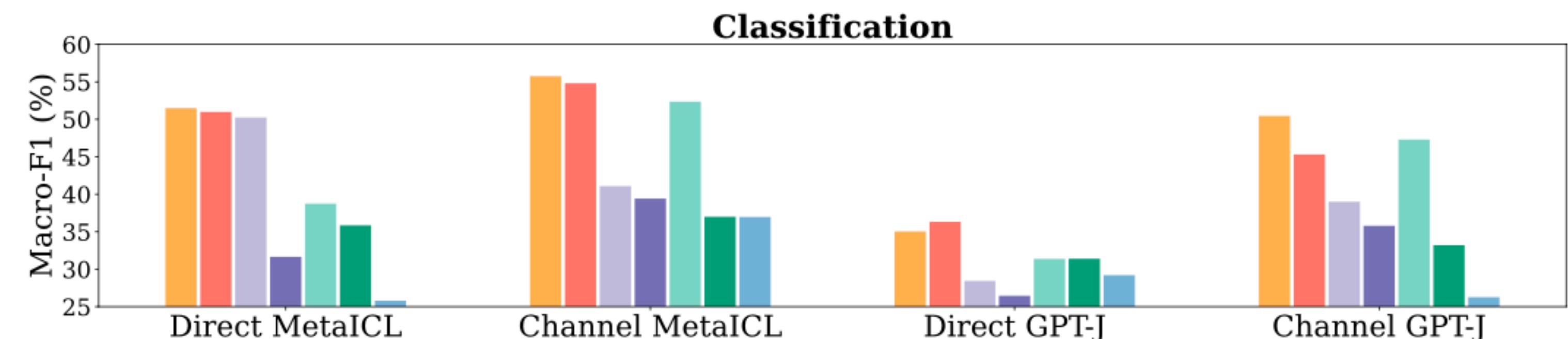
- 6 LLMs * 2 inference methods
 - normal & noisy channel
- 26 common tasks
 - classification & multiple choice
- different prompting methods
 - zero-shot
 - k-shot w/ correct labels
 - k-shot w/ scrambled labels



What makes in-context learning work?

Min et al. (2022)

- ▶ methods:
 - different prompting methods
 - zero-shot
 - k-shot w/ correct labels
 - k-shot w/ scrambled labels
 - k-shot w/ nonsense labels
 - k-shot w/ out-of-distribution inputs
 - k-shot w/ only input but no labels
 - k-shot w/ only labels but no input



	<i>F</i>	<i>L</i>	<i>I</i>	<i>M</i>
Gold labels	✓	✓	✓	✓
Random labels	✓	✓	✓	✗
OOD + Random labels	✓	✓	✗	✗
Random labels only	✗	✓	✗	✗
Random English words	✓	✗	✓	✗
No labels	✗	✗	✓	✗
No demonstrations	✗	✗	✗	✗

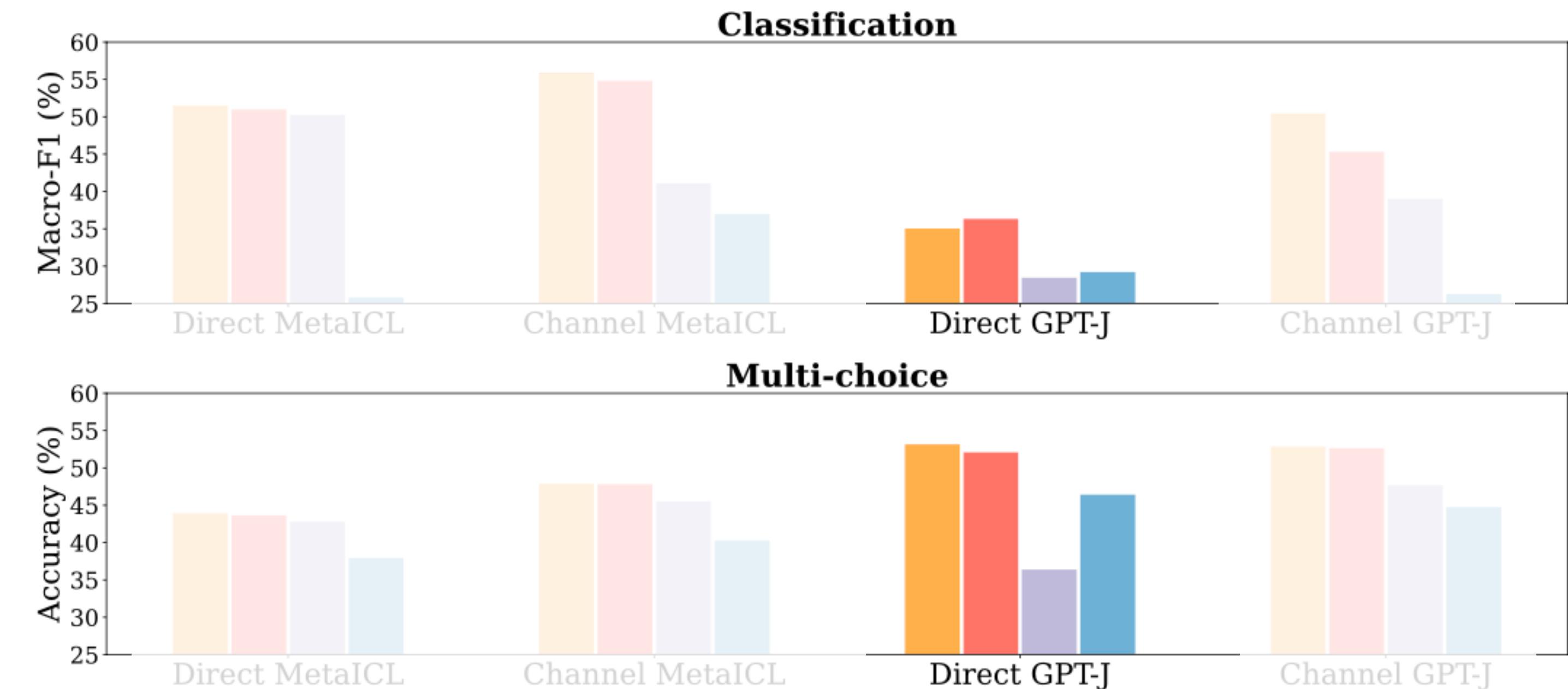
F: Format
L: Label space
I: Input distribution
M: Input-Label Mapping

What makes in-context learning work?

Min et al. (2022)

- ▶ in-distribution inputs matter
 - to standard models
- ▶ possible explanation:
 - the more similar the demonstrations are to the target input, the more the task resembles autoregressive text-generation; the task becomes “more similar” to what is seen in the demonstrations
 - unlike “meta-learning” sensu stricto

This suggests that in-distribution inputs in the demonstrations substantially contribute to performance gains. This is likely because conditioning on the in-distribution text makes the task closer to language modeling, since the LM always conditioned on the in-distribution text during training.



	<i>F</i>	<i>L</i>	<i>I</i>	<i>M</i>
Gold labels	✓	✓	✓	✓
Random labels	✓	✓	✓	✗
OOD + Random labels	✓	✓	✗	✗
No demonstrations	✗	✗	✗	✗

F: Format
L: Label space
I: Input distribution
M: Input-Label Mapping

Do models understand the meaning of the prompts?

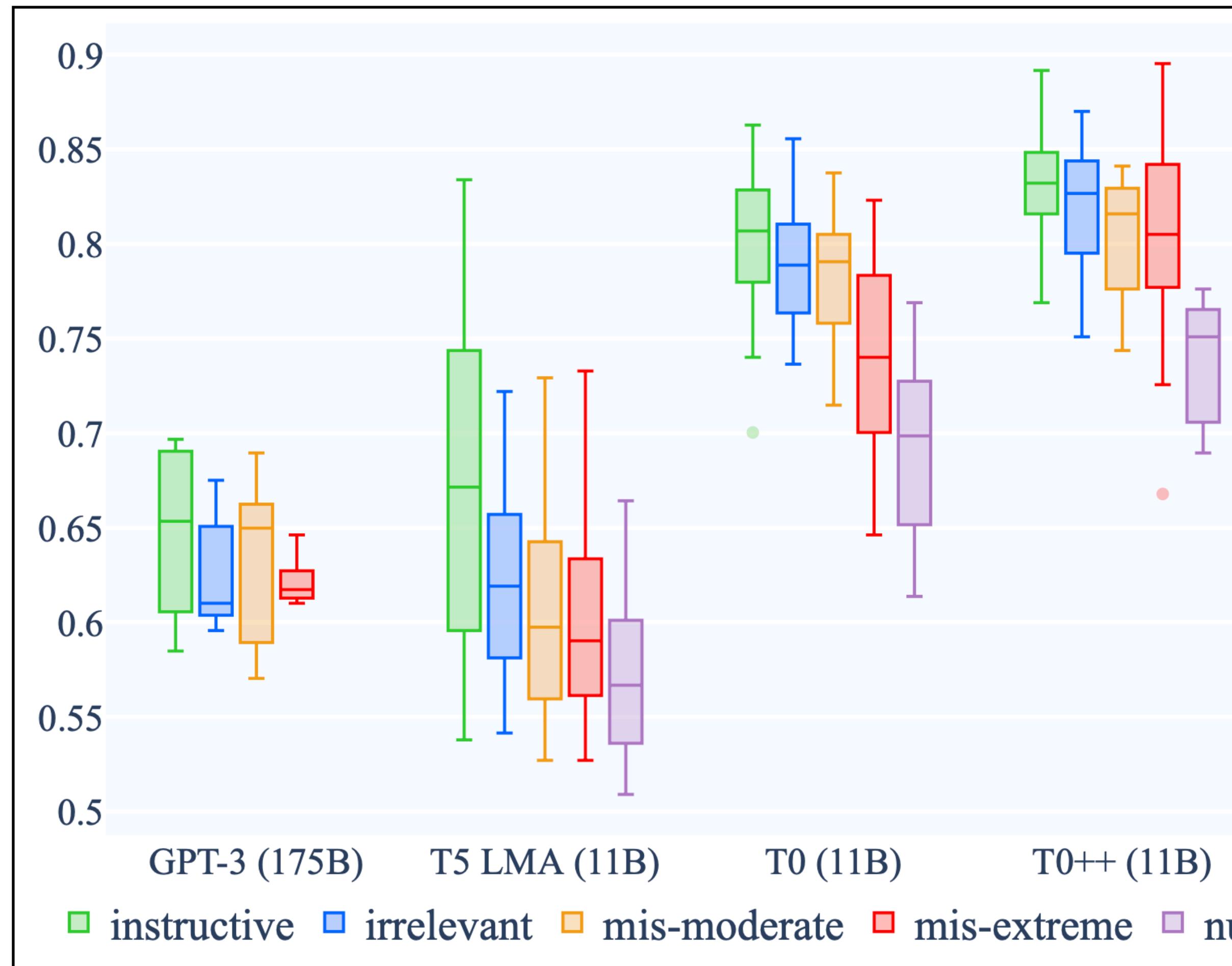
Webson & Pavlick (2022)

- ▶ zero-shot, and k -shot in-context learning
 - $k \in \{0, 4, 8, 16, 32, 64, 128, 256\}$
- ▶ different example templates
 - see →
- ▶ different target words
 - yes/no
 - “yes/no”-like (true/false, positive/negative)
 - arbitrary (cat/dog, Jane/Jake)
 - reversed (no/yes)

Category	Examples
instructive	{prem} Are we justified in saying that “{hypo}”? Suppose {prem} Can we infer that “{hypo}”?
misleading-moderate	{prem} Can that be paraphrased as: “{hypo}”? {prem} Are there lots of similar words in “{hypo}”?
misleading-extreme	{prem} is the sentiment positive? {hypo} {prem} is this a sports news? {hypo}
irrelevant	{prem} If bonito flakes boil more than a few seconds the stock becomes too strong. "{hypo}"?
null	{premise} {hypothesis} {hypothesis} {premise}

Do models understand the meaning of the prompts?

Webson & Pavlick (2022)

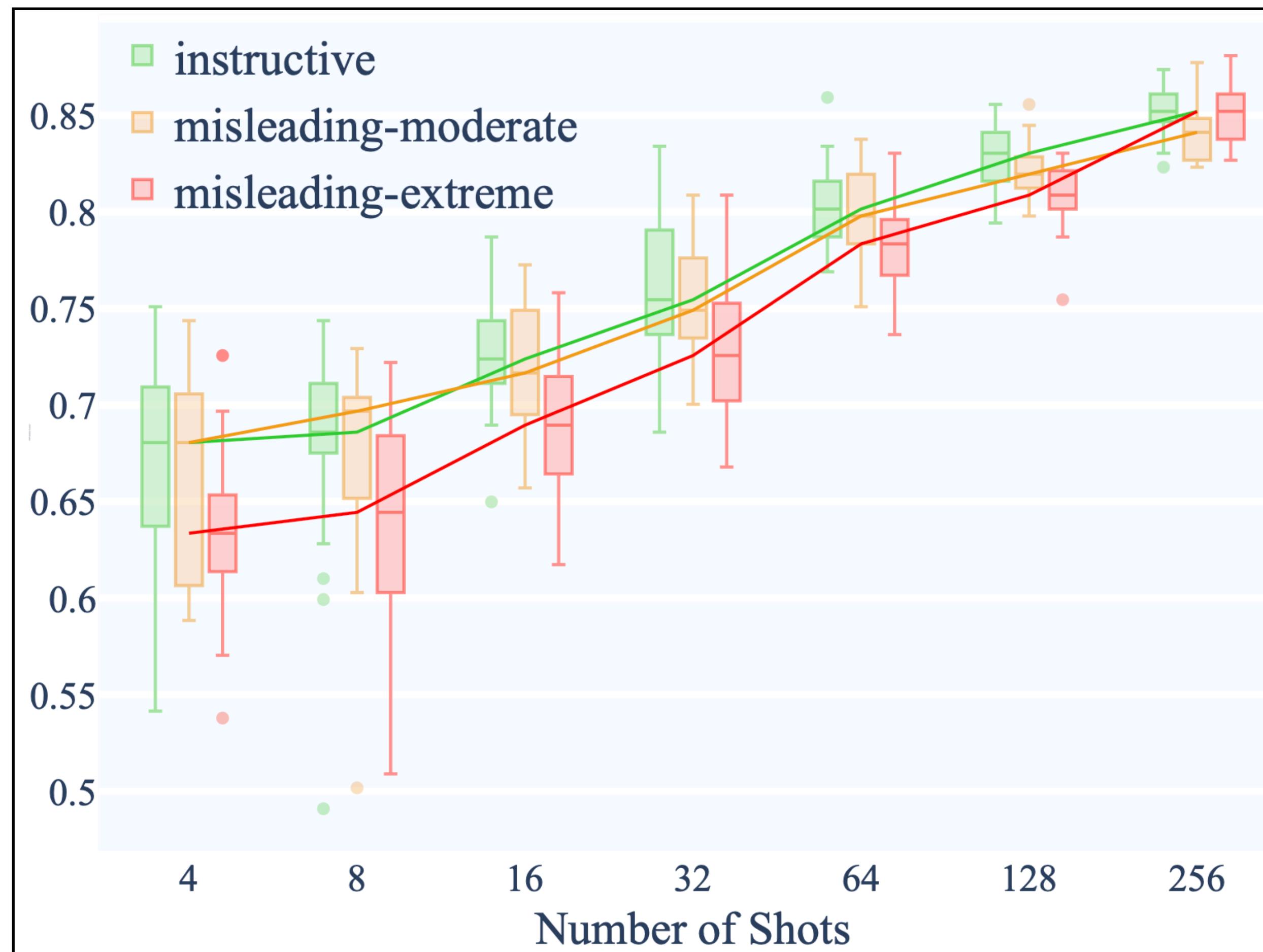


Category	Examples
instructive	{prem} Are we justified in saying that "{hypo}"? Suppose {prem} Can we infer that "{hypo}"?
misleading-moderate	{prem} Can that be paraphrased as: "{hypo}"? {prem} Are there lots of similar words in "{hypo}"?
misleading-extreme	{prem} is the sentiment positive? {hypo} {prem} is this a sports news? {hypo}
irrelevant	{prem} If bonito flakes boil more than a few seconds the stock becomes too strong. "{hypo}"?
null	{premise} {hypothesis} {hypothesis} {premise}

16-shot accuracy: no differences btw.
categories except for comparisons against “null”

Do models understand the meaning of the prompts?

Webson & Pavlick (2022)



Category	Examples
instructive	{prem} Are we justified in saying that "{hypo}"? Suppose {prem} Can we infer that "{hypo}"?
misleading-moderate	{prem} Can that be paraphrased as: "{hypo}"? {prem} Are there lots of similar words in "{hypo}"?
misleading-extreme	{prem} is the sentiment positive? {hypo} {prem} is this a sports news? {hypo}
irrelevant	{prem} If bonito flakes boil more than a few seconds the stock becomes too strong. "{hypo}"?
null	{premise} {hypothesis} {hypothesis} {premise}

performance of T0 (3B): only difference btw.
“misleading extreme” for 8 to 128 shots

Do models understand the meaning of the prompts?

Webson & Pavlick (2022)

- ▶ different target words
 - yes/no
 - “yes/no”-like (true/false, positive/negative)
 - arbitrary (cat/dog, Jane/Jake)
 - reversed (no/yes)
- ▶ target word matters for average success
- ▶ but unintuitive interactions with:
 - the former can outperform the latter
 - {premise} Does the paragraph start with “the”? {hypothesis} [yes/no]
 - {premise} Based on the previous passage, is it true that “{hypothesis}”? [cat/dog]

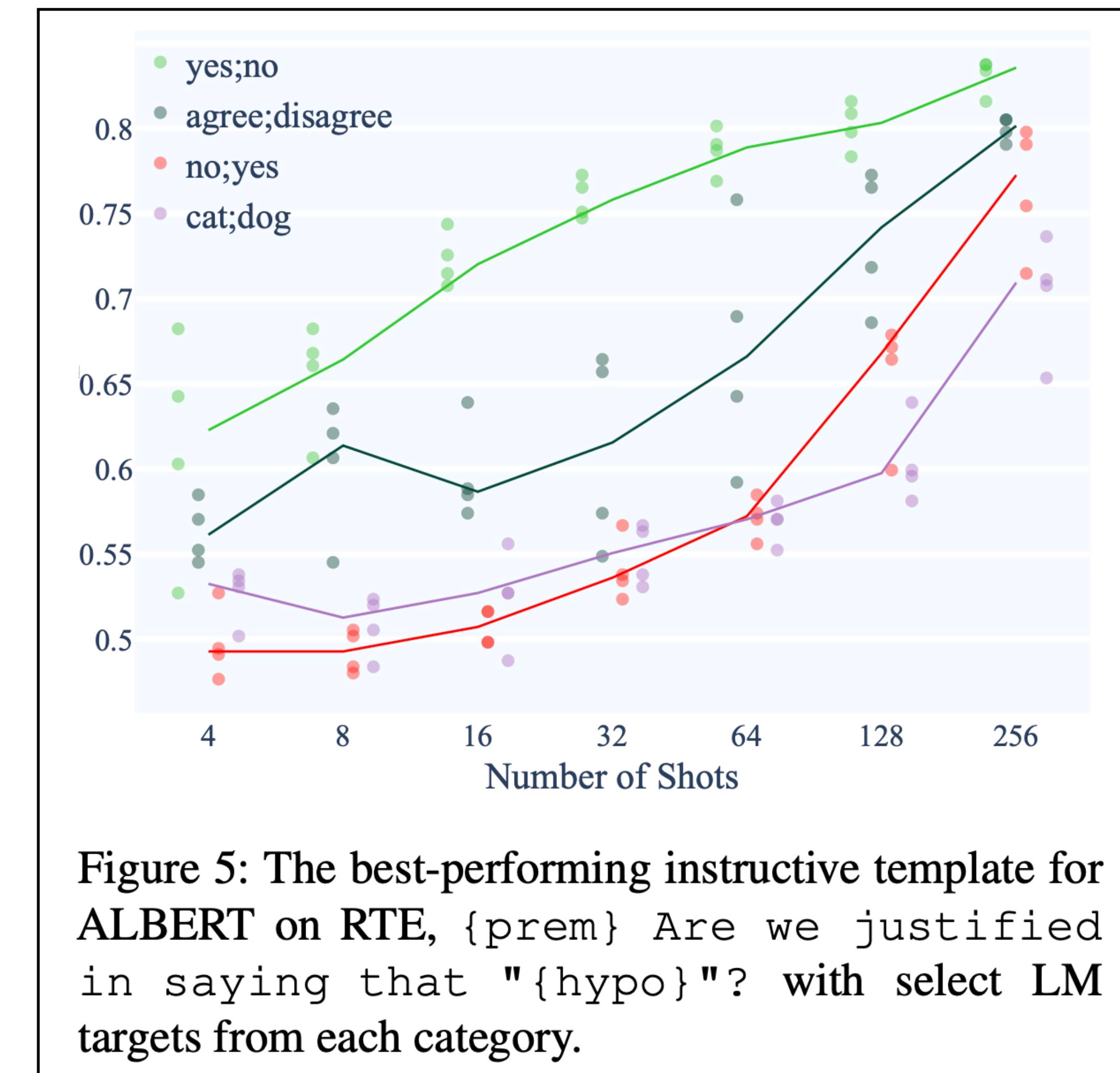


Figure 5: The best-performing instructive template for ALBERT on RTE, {prem} Are we justified in saying that “{hypo}”? with select LM targets from each category.

Can LMs learn from explanations in context?

Lampinen et al. (2022)

- ▶ **method:**
 - supplement ICL with additional examples
- ▶ **main findings:**
 - explanations **can** further improve ICL performance
 - hand-tuned explanations work better
 - benefit only for larger LMs

Task instruction { Answer these questions by identifying whether the second sentence is an appropriate paraphrase of the first, metaphorical sentence.

Few-shot example #1 { Q: David's eyes were like daggers at Paul when Paul invited his new girlfriend to dance. <-> David had two daggers when Paul invited his new girlfriend to dance.
choice: True
choice: False
A: False

Answer explanation { Explanation: David's eyes were not literally daggers, it is a metaphor used to imply that David was glaring fiercely at Paul.

4 more examples
+ explanations

⋮

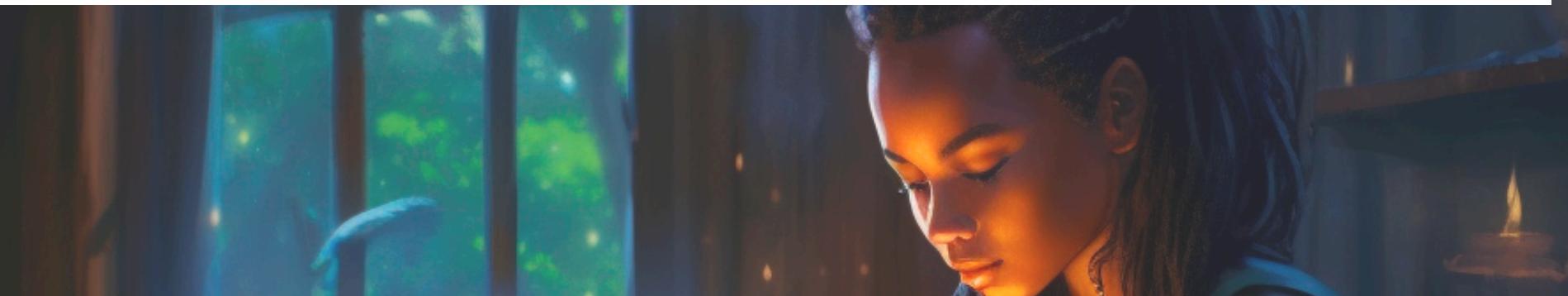
Target question { Q: Our whole life we swim against the waves towards the green light of happiness. <-> Our whole life we try to reach happiness.
choice: True
choice: False
A:

Summary

In-context learning

- ▶ structure of simple prompt also matters; so ICL need not be genuine “learning”
 - Reynolds & McDonell (2021)
- ▶ k-shot examples may provide statistical cues which the LM can exploit
 - Xie et al. (2022)
- ▶ similarity of demonstrations and target matters
 - Min et al. (2022)
- ▶ instructions and label correctness do not matter
 - Min et al. (2022), Webson & Pavlick (2022)
- ▶ still, good explanations can help (for large LMs)
 - Lampinen et al. (2022)

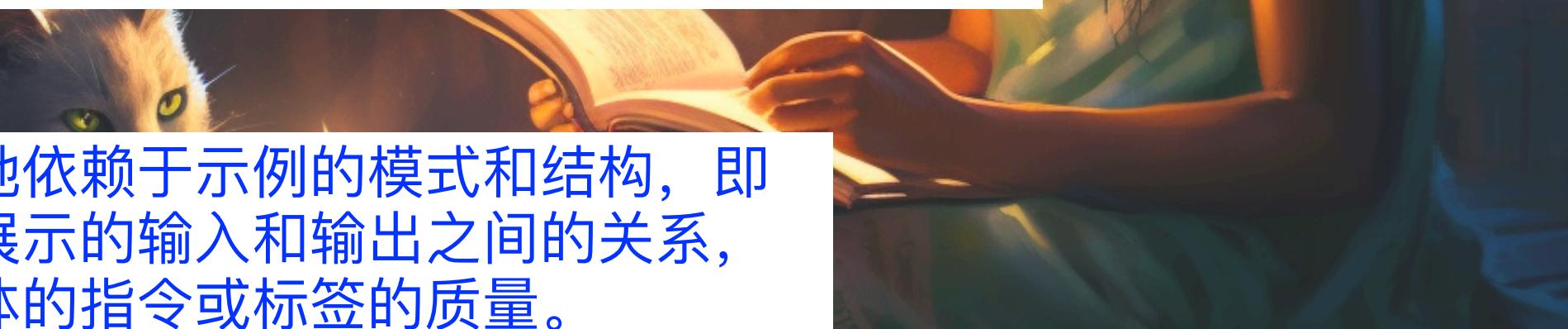
在上下文学习 (In-context Learning) 中, 尤其是在提示学习 (Prompt Learning) 中, 模型通过给定的上下文 (如几个示例或提示) 来直接生成结果, 而不涉及参数更新或显式的训练过程。因此, 这种形式的学习不完全符合“传统意义上的学习过程”的定义, 而更像是一种即时的模式匹配或统计推断。



提供几个示例 (k-shot) 可以让语言模型从中提取统计规律, 从而提高表现。



训练示例与目标任务的相似性会影响模型的学习效果。

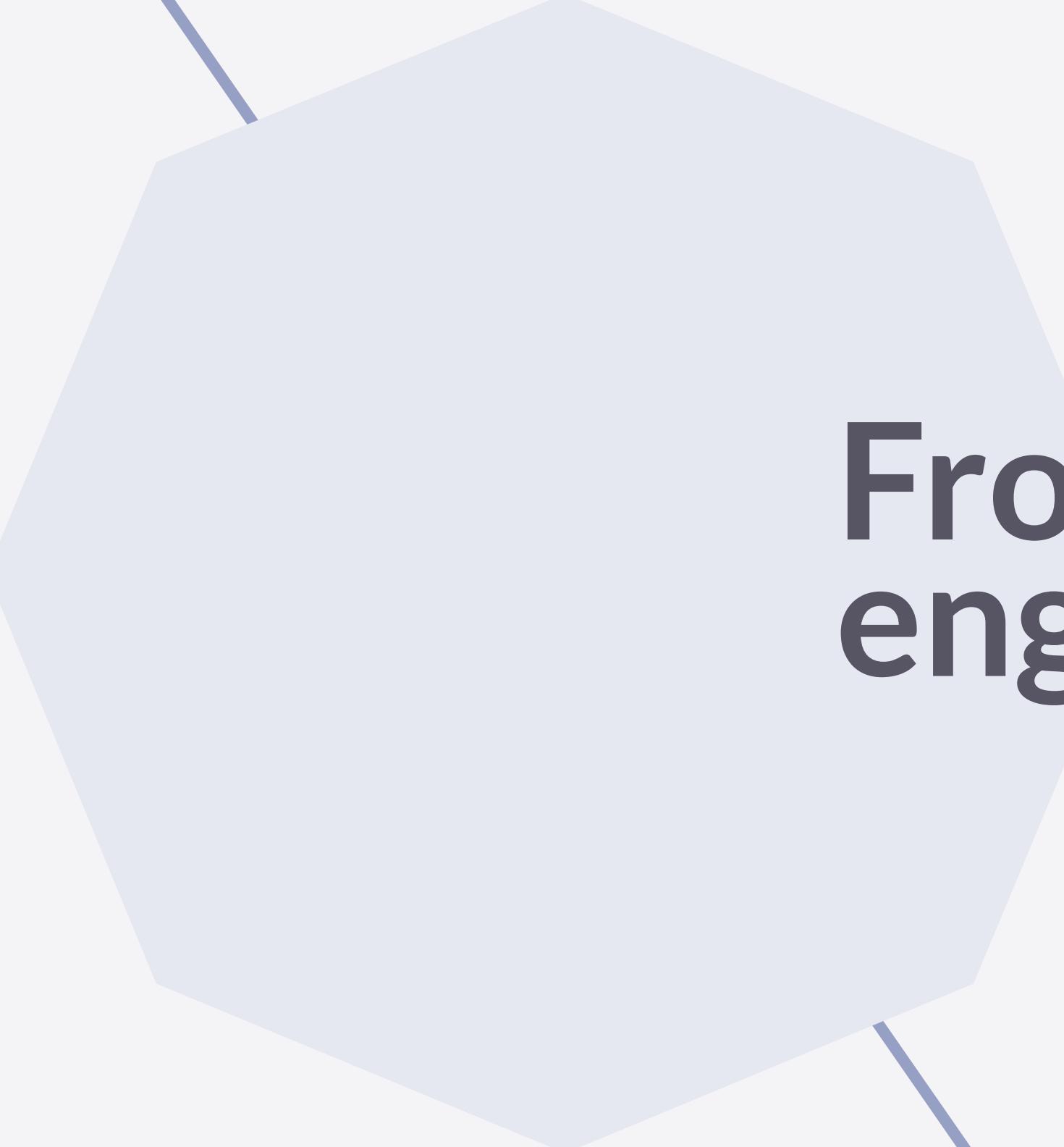


模型更多地依赖于示例的模式和结构, 即示例中所展示的输入和输出之间的关系, 而不是具体的指令或标签的质量。



尽管指令和标签的正确性不重要, 清晰且详细的解释仍能帮助大型语言模型更好地理解和完成任务。





**From simple to
engineered prompting**

Counting letters

Exploring step-by-step reasoning

INPUT

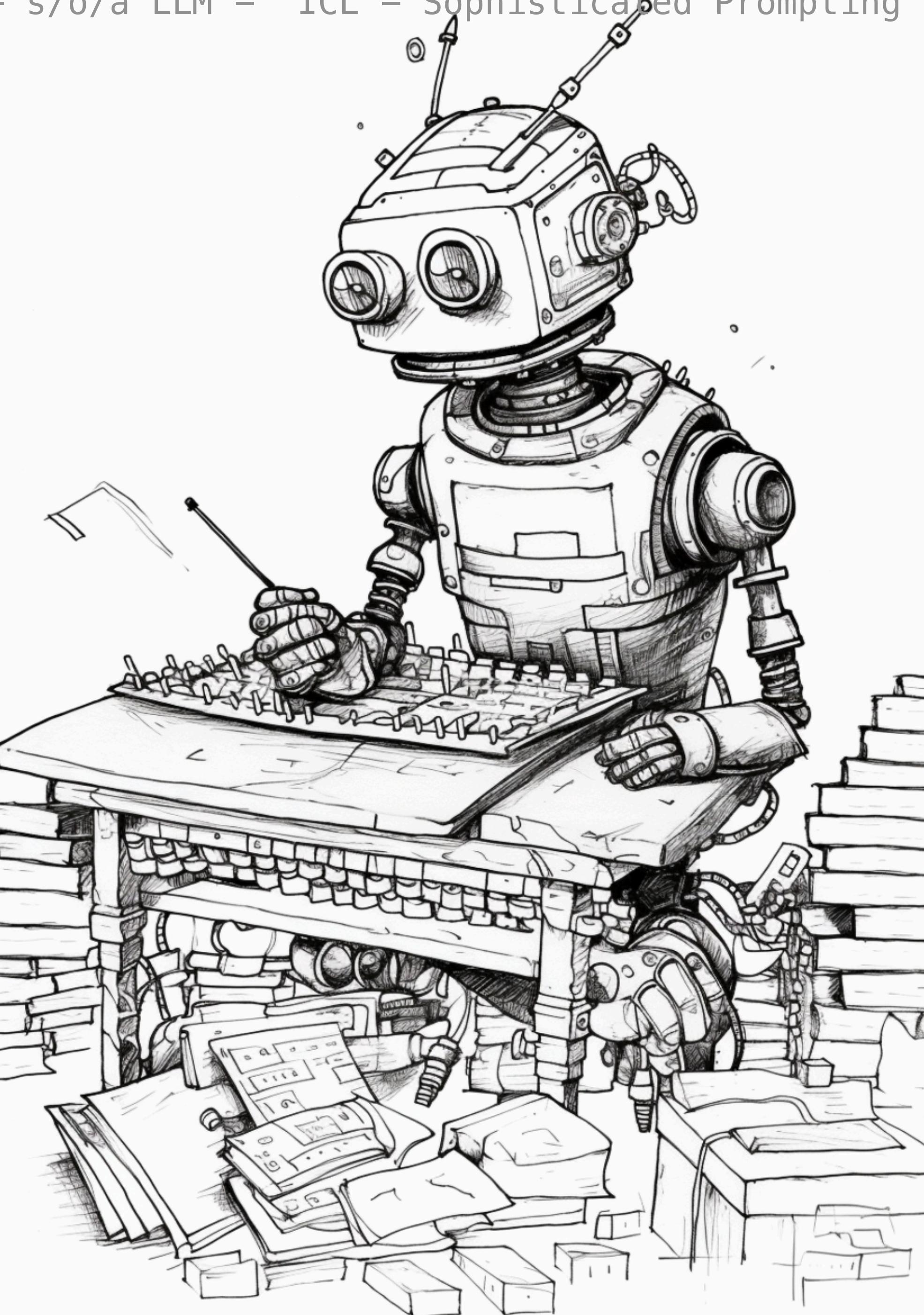
Do the numbers of letters in all words starting with a vowel from the following list sum up to 42?

Polina, Michael, eggplant, cheese, oyster, imagination, elucidation, induce

Please answer just 'yes' or 'no'

OUTPUT

>>> ???



demo



Structured reasoning

GPT-3.5 turbo (Dec 2023)

Do the numbers of letters in all words starting with a vowel from the following list sum up to 42?

Polina, Michael, eggplant, cheese, oyster, imagination, elucidation, induce

Please answer just 'yes' or 'no'

No

No = 98.42%

Yes = 1.57%

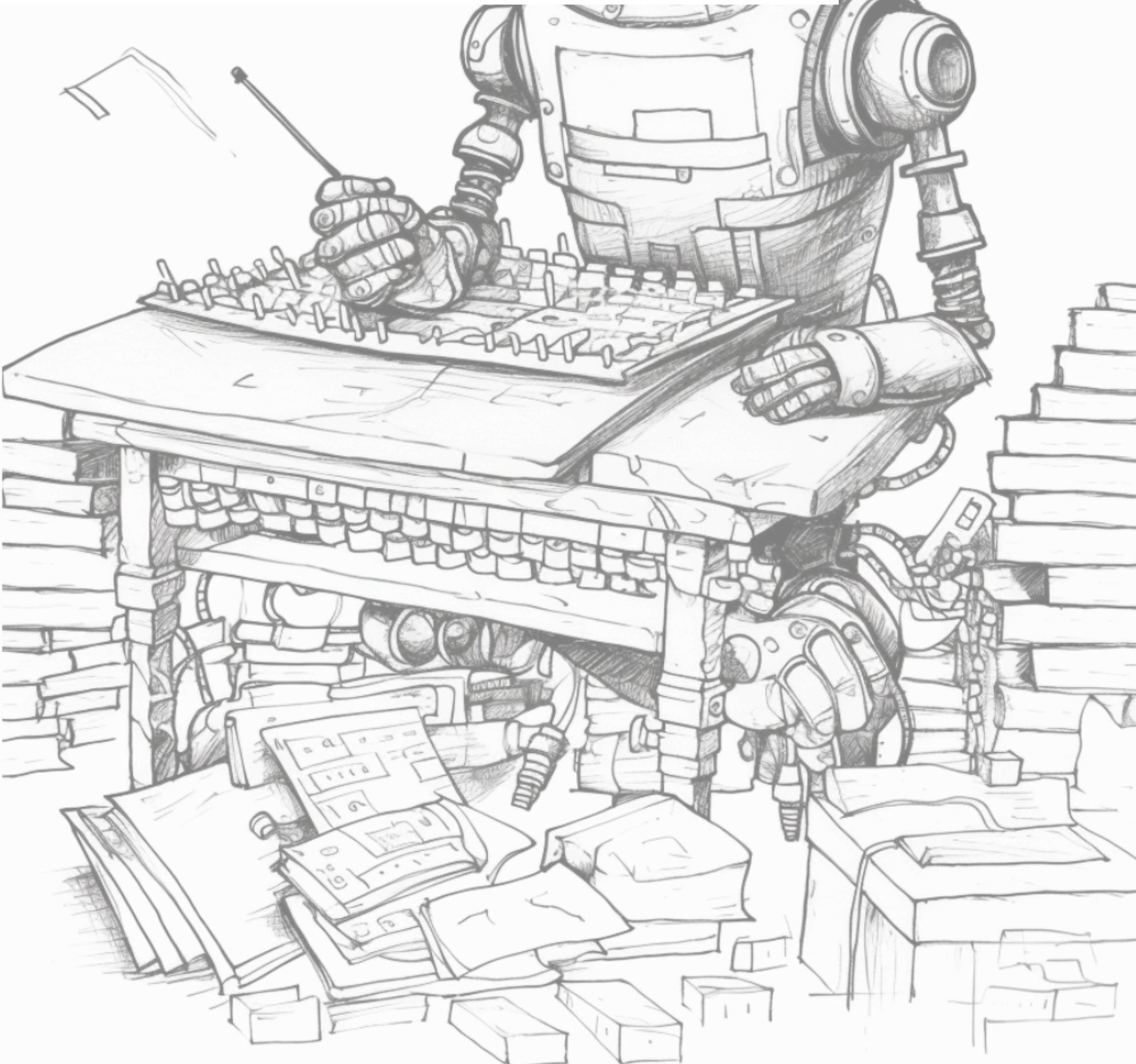
NO = 0.01%

no = 0.01%

No = 0.00%

Total: -0.02 logprob on 1 tokens
(100.00% probability covered in top 5 logits)

指通过明确的步骤和逻辑关系进行系统化思考和问题解决的过程。这种推理方式注重步骤的顺序和每一步之间的关系，以确保结论的准确性和可靠性。



Structured reasoning

GPT-3.5 turbo (Dec 2023)

Do the numbers of letters in all words starting with a vowel from the following list sum up to 42?

Polina, Michael, eggplant, cheese, oyster, imagination, elucidation, induce

Let's think step by step. Correctness is very important here.

We can quickly observe that Polina, eggplant, oyster, and imagination all start with a vowel.

Letters in Polina: 6

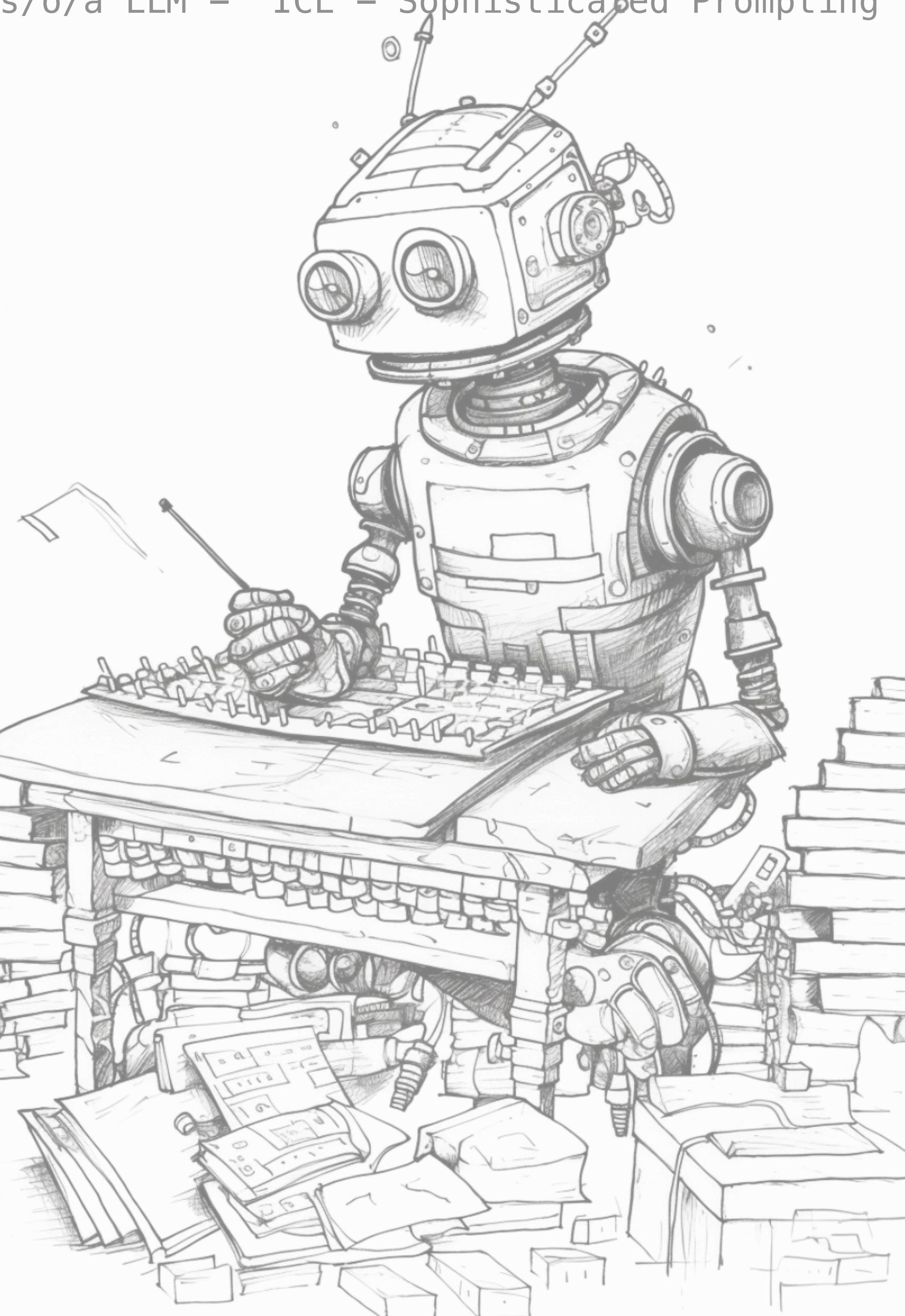
Letters in eggplant: 8

Letters in oyster: 6

Letters in imagination: 11

$$6 + 8 + 6 + 11 = 31$$

The answer is no, the numbers of letters in all words starting with a vowel from the list do not sum up to 42.



Structured reasoning

ChatGPT (Dec 2023)



You

Do the numbers of letters in all words starting with a vowel from the following list sum up to 42?

Polina, Michael, eggplant, cheese, oyster, imagination, elucidation, induce

Please answer just 'yes' or 'no'



ChatGPT

No.



You

Are you sure?



ChatGPT

Yes, the numbers of letters in all words starting with a vowel from the list do sum up to 42. [→]

Analysis

python

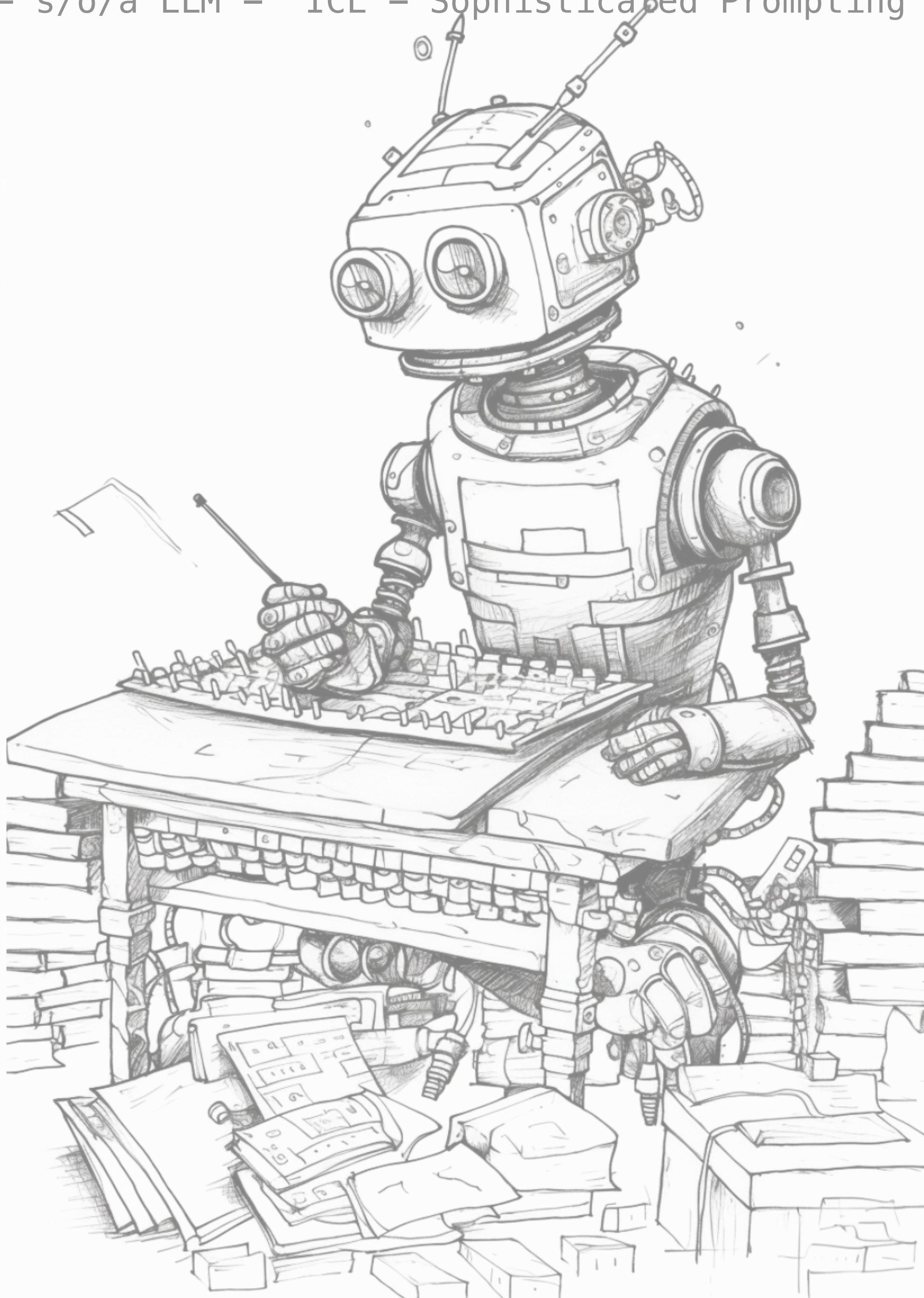
Copy code

```
# List of words starting with a vowel
words = ["eggplant", "oyster", "imagination", "elucidation"]

# Calculating the sum of the number of letters in each word
total_letters = sum(len(word) for word in words)
total_letters == 42
```

Result

True



Chain-of-Thought prompting

- ▶ give task instruction
- ▶ give one or more **examples with explicit chain-of-thought reasoning leading to the correct answer**
- ▶ works for example to complex for few-shot prompting
- ▶ requires “right” task analysis in CoT steps

INPUT

The odd numbers in this group add up to an even number:
4, 8, 9, 15, 12, 2, 1.

A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.

The odd numbers in this group add up to an even number:
15, 32, 5, 13, 82, 7, 1.

A:

OUTPUT

Adding all the odd numbers (15, 5, 13, 7, 1) gives 41.
The answer is False.

Zero-shot CoT prompting

► just add “Let’s think step by step”

- even better (Zhou et al. 2022): "Let's work this out in a step by step way to be sure we have the right answer."

(a) Few-shot

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: 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 answer is 8. ✗

(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. ✓

(c) Zero-shot

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: The answer (arabic numerals) is

(Output) 8 ✗

(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. ✓

INPUT

The odd numbers in this group add up to an even number:
15, 32, 5, 13, 82, 7, 1.

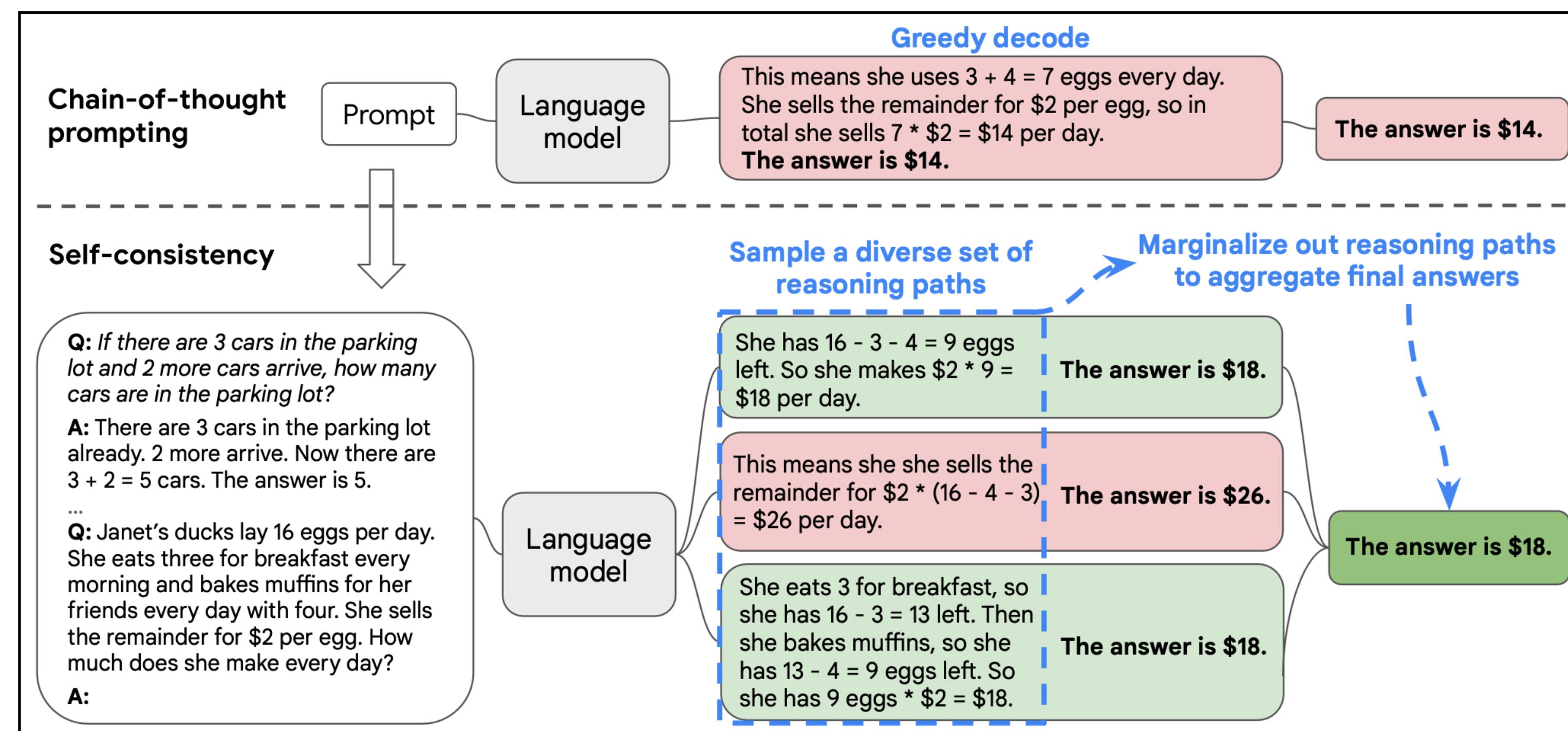
A: Let's think step by step.

OUTPUT

Adding all the odd numbers (15, 5, 13, 7, 1) gives 41.
The answer is False.

Self-consistency prompting

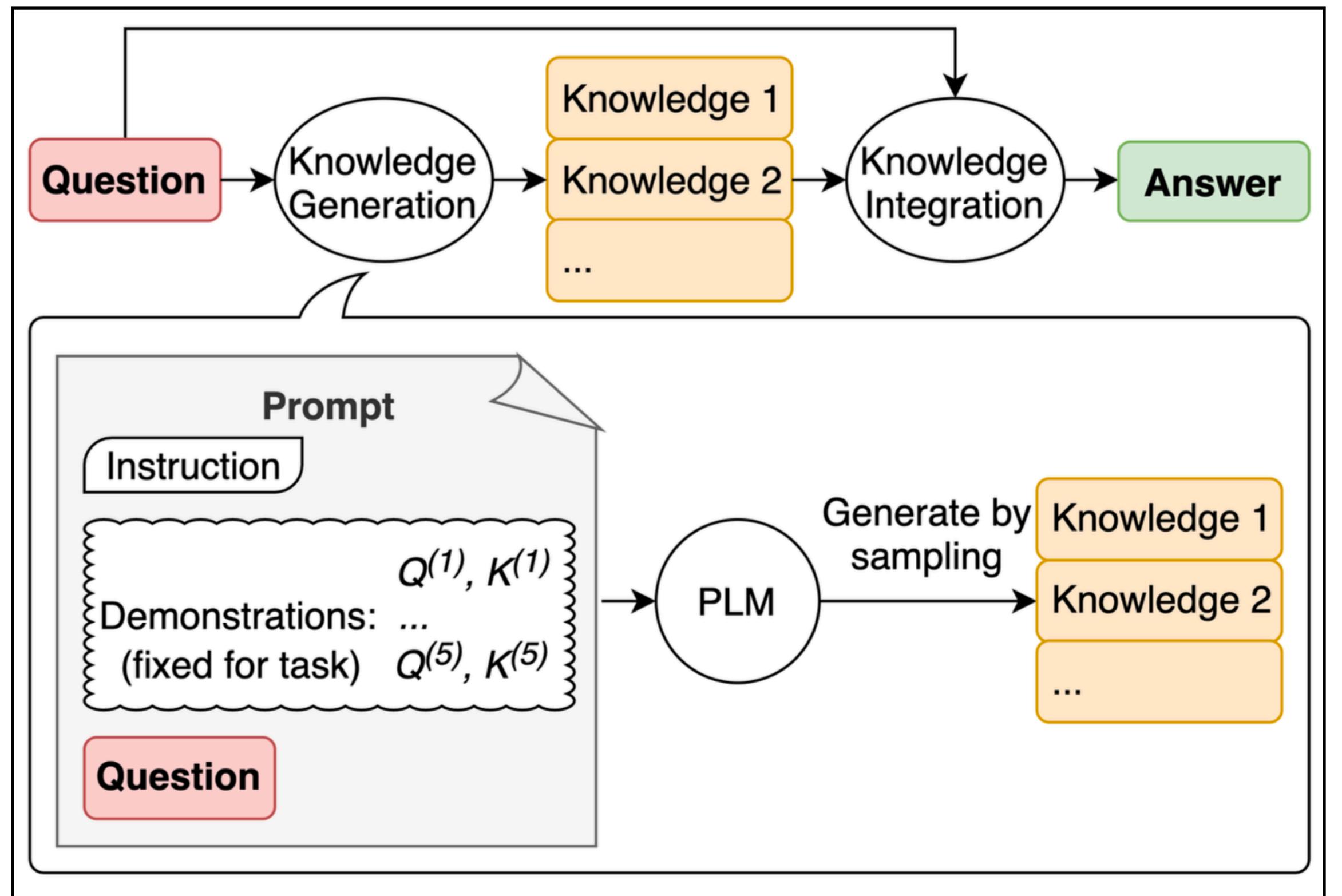
- ▶ few-shot CoT prompting with self-generated CoT sequences (greedily)
- ▶ aggregation over stochastic answer generation



Generated knowledge prompting

for common sense QA

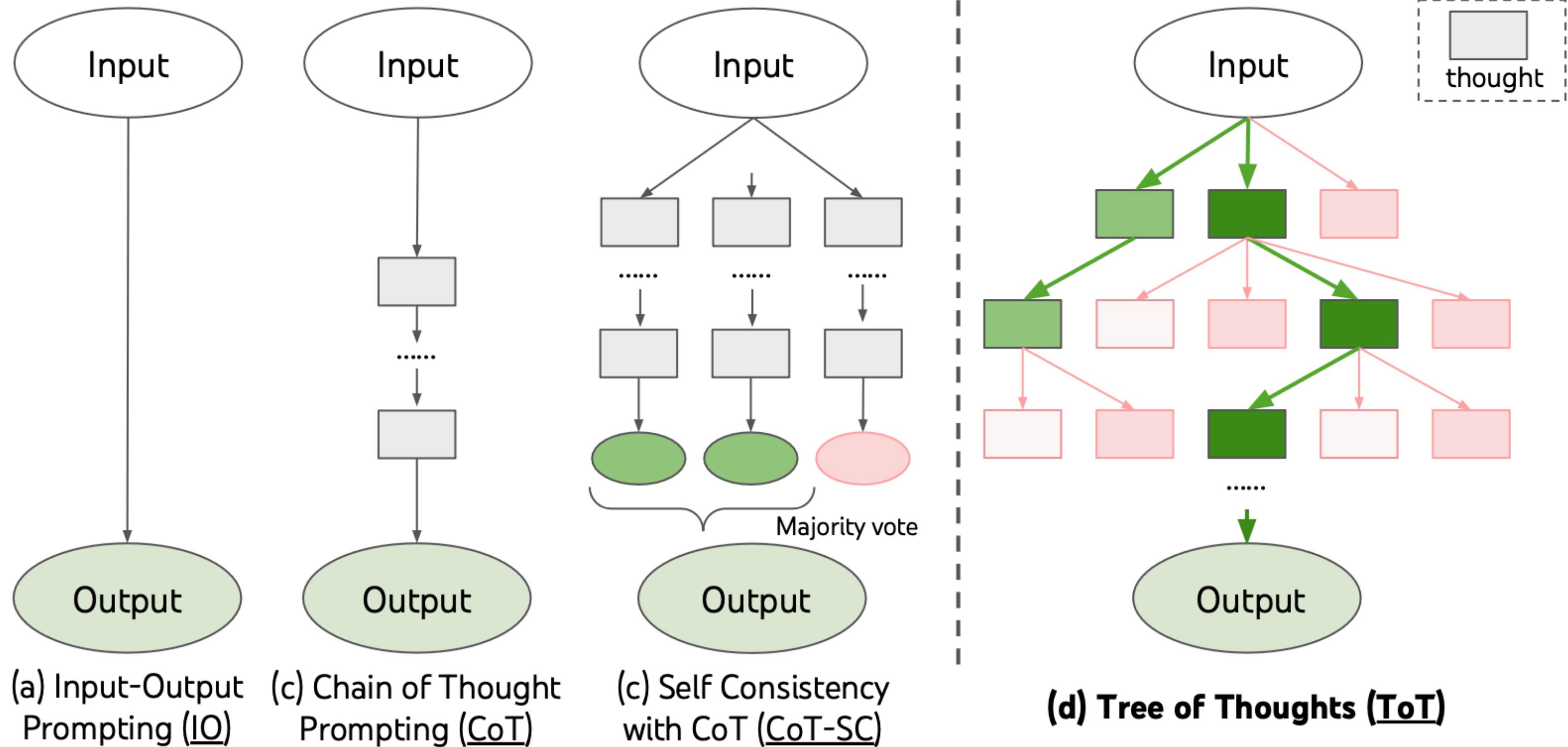
- ▶ generate knowledge statements K for question Q
- ▶ generate several answers A 's to Q for each K
- ▶ final answer to Q is max of weighted A 's



Tree of Thought

deliberate problem solving with LLMs

motivation: LLM是token-level的推理，从左到右，对比人类思考方式局限性大：1/局部性：不会在思维过程中探索当前节点的其它分支解决方案 2/缺乏全局性：未纳入任何类型的规划、展望、回溯来帮助评估不同选项。



ToT: 可以有多个选择路径，可以进行剪枝、回溯、自我评估，最终找到最优答案路径进行输出。

Tree of Thought

modules of reasoning

4个步骤: thought分解 – thought生成 – 状态评估 – 搜索算法

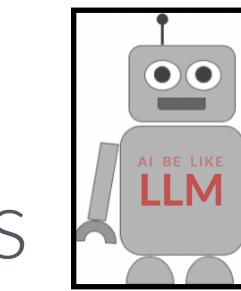
▶ thought decomposition

- what counts as a next step?



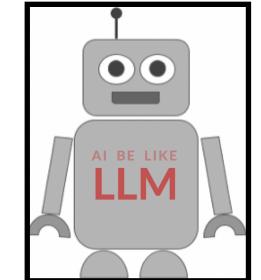
▶ thought generation

- sample / generate next steps



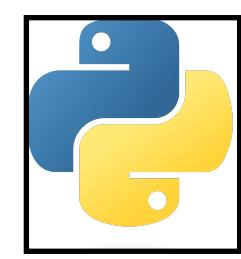
▶ state evaluation

- evaluate / vote states based on closeness to goal

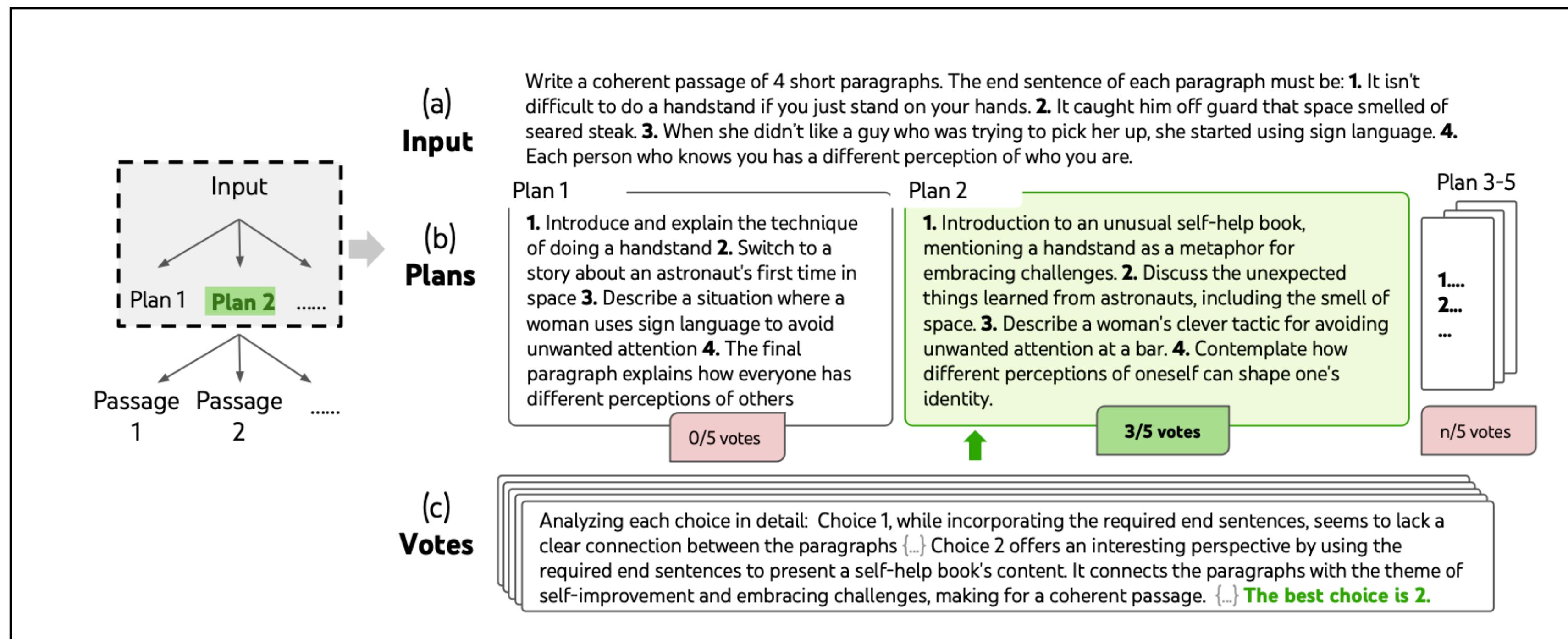


▶ search algorithm

- how to build / traverse the tree



https://blog.csdn.net/qq_27590277/article/details/131448071



Creative writing task

Chain of Thought Prompting:

Encourages step-by-step reasoning to reach an answer.

Zero-Shot Prompting:

Provides an answer without any examples or additional context.

Self-Consistency Prompting:

Generates multiple responses and selects the most consistent one.

Generated-Knowledge Prompting:

Uses generated background knowledge to inform the response.

Tree-of-Thought:

Explores multiple reasoning paths to find the best solution.

Zero-Shot Prompting with CoT:

Combines zero-shot prompting with step-by-step reasoning.

Few-Shot Prompting:

Provides a few examples to guide the response.

Few-Shot Prompting with CoT:

Combines few-shot examples with step-by-step reasoning.

我们以解决一个简单的数学问题为例：求解 $23 + 19$

Chain of Thought Prompting (思维链提示)：

"首先把23和19分成个位和十位部分，23分成20和3，19分成10和9，然后先加十位部分， $20 + 10 = 30$ ，然后加个位部分， $3 + 9 = 12$ ，最后再把两部分相加， $30 + 12 = 42$ 。"

Zero-Shot Prompting (零样本提示)：

" $23 + 19 = 42$ "

Self-Consistency Prompting (自一致性提示)：

" $23 + 19$ 等于42，因为我们可以分解为 $20 + 3$ 和 $10 + 9$ ，然后分别相加再合并结果。多个步骤都验证这一计算是正确的。"

Generated-Knowledge Prompting (生成知识提示)：

"根据算术规则，加法是将两个数值进行求和，23和19相加结果是42。"

Tree-of-Thought (思维树)：

"我们可以从多个角度来解决这个问题。首先，把23和19分解成简单的部分，比如 $20 + 3$ 和 $10 + 9$ ，然后分别相加。或者直接从右往左逐位相加，这两种方法都得出42。"

Zero-Shot Prompting with CoT (零样本提示加思维链)：

"求解 $23 + 19$ 时，先考虑个位和十位部分，然后分别相加，得出最终结果42。"

Few-Shot Prompting (少样本提示)：

"例如， $12 + 8 = 20$, $35 + 15 = 50$ ，类似地， $23 + 19 = 42$ 。"

Few-Shot Prompting with CoT (少样本提示加思维链)：

"例如， $12 + 8$ 可以分解为 $10 + 2$ 和8，然后相加得20； $35 + 15$ 可以分解为 $30 + 5$ 和 $10 + 5$ ，然后相加得50。同理， $23 + 19$ 可以分解为 $20 + 3$ 和 $10 + 9$ ，然后相加得42。"

Summary

prompt engineering

- ▶ different kinds of prompting techniques:
 - zero-shot w/ or w/o CoT
 - few-shot w/ or w/o CoT
 - ensemble methods
 - self-consistency
 - generated knowledge prompting
 - task-composition / neuro-symbolic generation:
 - tree of thoughts

