ML Collective

Deep Learning Classics and Trends

21 April 2023

Augmented Language Models : A Survey

Based on

Mialon, Grégoire, et al. "Augmented language models: a survey." https://arxiv.org/abs/2302.07842

MS, University of Illinois at Urbana Champaign (UIUC)

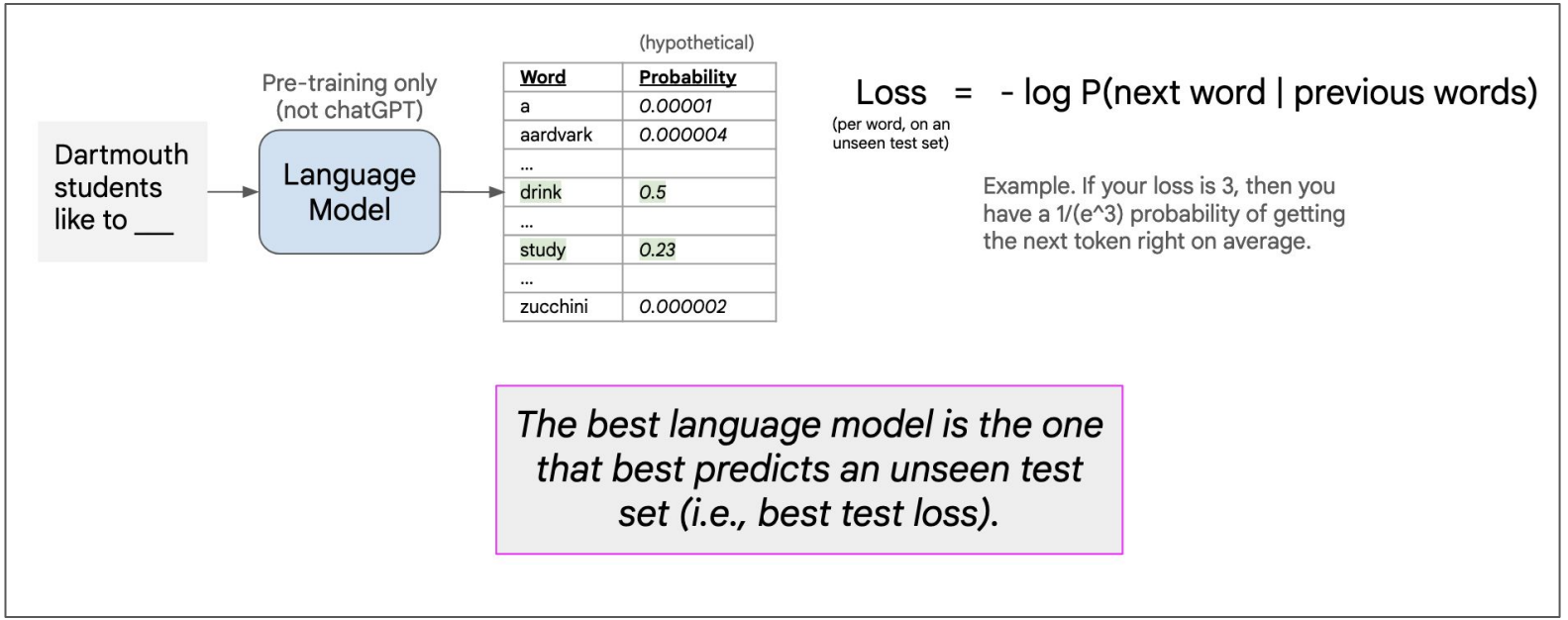
Gargi Balasubramaniam

If you find this useful, please consider citing:

Balasubramaniam, Gargi. "Presentation on Augmented Language Models: A Survey." (April 2023).
https://github.com/bgargi/aug-lang-models-pres

```
@misc{balasubramaniam23ALM,
  author    = {Balasubramaniam, Gargi},
  title     = {Presentation on {Augmented Language Models: A Survey}},
  year      = {2023},
  month     = {April},
  howpublished = {PDF slides},
  note      = {Based on the survey paper by Mialon, Gr{é}goire, et al. {"Augmented Language Models: a Survey"} \url{https://arxiv.org/abs/2302.07842}},
  url       = {https://github.com/bgargi/aug-lang-models-pres},
}
```

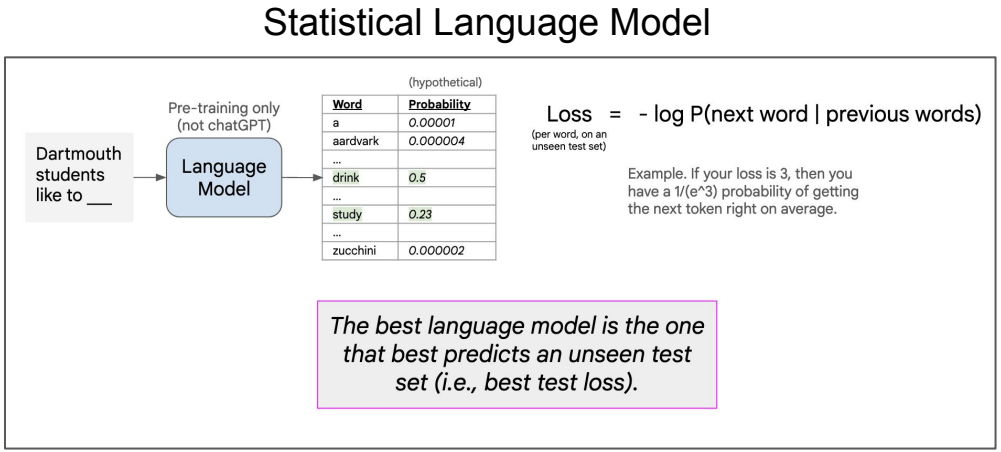
Review



Source

Why?

Why might one need to augment language models, and how?

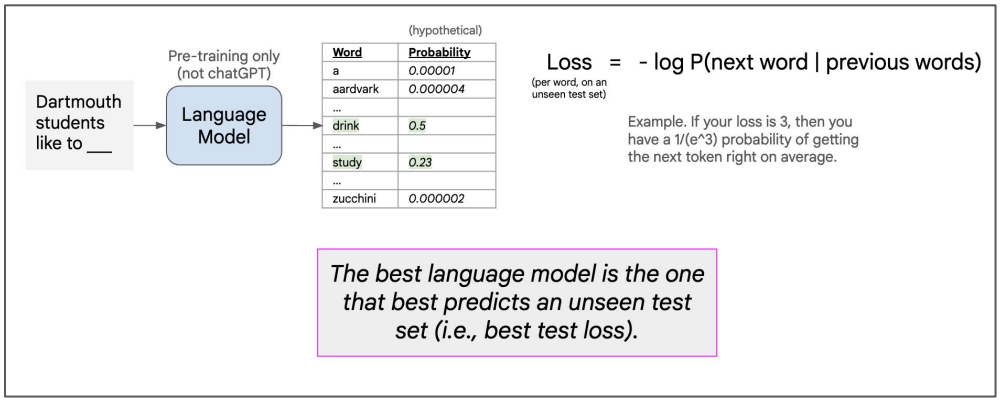


Why?

Why might one need to augment language models, and how?

Single Parametric Model

Limited Context



What is possible

What do language models learn from next-word prediction?

Grammar	In my free time, I like to {**run**, banana}
Lexical semantics	I went to the zoo to see giraffes, lions, and {**zebras**, spoon}
World knowledge	The capital of Denmark is {**Copenhagen**, London}
Sentiment analysis	Movie review: I was engaged and on the edge of my seat the whole time. The movie was {**good**, bad}
Harder sentiment analysis	Movie review: Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was {**bad**, good}
Translation	The word for “pretty” in Spanish is {**bonita**, hola}
Spatial reasoning	[...] Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the {**kitchen**, store}
Math question	First grade arithmetic exam: 3 + 8 + 4 = {**15**, 11}

[thousands (millions?) more]

Extreme multi-task learning!

Source

.. and what is not

What can't language models learn from next-word prediction?

Current world knowledge	The stock price of APPL on March 1st, 2023 is {???}
Arbitrarily long arithmetic	36382894730 + 238302849204 = {???}
Many-step reasoning	Take the nineteenth digit of Pi and multiply it by the e to the fourth power. The resulting ones-digit of the resulting number is {???}
Predict the future	The winner of the FIFA world cup in 2026 is {???}
Information not in the training data	Jason Wei's favorite color is {???}
Extremely long inputs	[2,000 page Harry Potter fan-fiction] What happened after Harry opened the chest for the second time? {???}

Source

Why?

Why might one need to augment language models, and **how**?

Single Parametric Model

Limited Context

Statistical Language Model

Today

Induce reasoning

Allow usage of tools

Leverage external data sources (e.g. retrieve documents from a dataset)

Key: (clever) context expansion

Disclaimer

- What comes to your mind when you think of “reasoning” in the context of language models?

Disclaimer

- “Reasoning” = decomposing into simpler subtasks and solving those legitimately to obtain a correct answer
  - Simply producing a larger context to correctly produce the missing tokens? Perhaps.

model	date	developer	# parameters
GPT-3	May 2020	OpenAI	175,000,000,000
GPT-Neo	Mar 2021	EleutherAI	2,700,000,000
GPT-J	Jun 2021	EleutherAI	6,000,000,000
Megatron-Turing NLG	Oct 2021	Microsoft & Nvidia	530,000,000,000
Gopher	Dec 2021	DeepMind	280,000,000,000
LaMDA	Jan 2022	Google	137,000,000,000
GPT-NeoX	Feb 2022	EleutherAI	20,000,000,000
Chinchilla	Mar 2022	DeepMind	70,000,000,000
PaLM	Apr 2022	Google	540,000,000,000
Luminous	Apr 2022	Aleph Alpha	70,000,000,000
OPT	May 2022	Meta	175,000,000,000
BLOOM	Jul 2022	Hugging Face collaboration	175,000,000,000
GPT-3.5	Nov 2022	OpenAI	Unknown
LLaMA	Feb 2023	Meta	65,000,000,000
GPT-4	Mar 2023	OpenAI	Unknown

Source

Not all language models are large.. :)

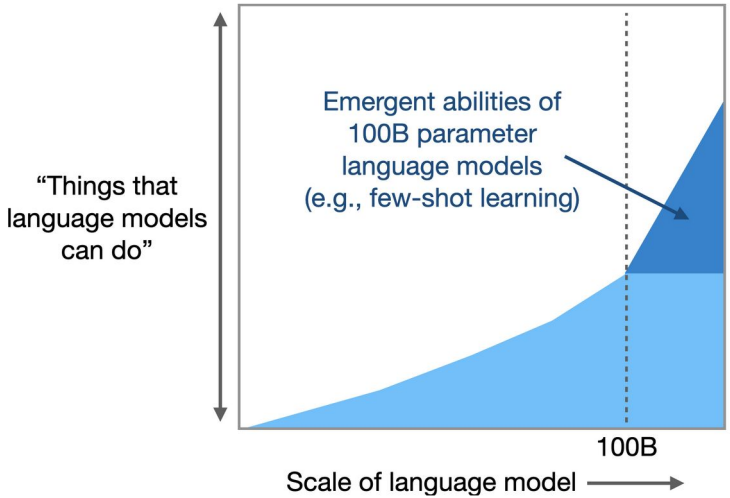
What does scaling mean and how is it measured?

$\propto$ x Model size x Training tokens = Training compute

	Model size (# parameters)	Training data (# tokens)	Training compute (FLOPs)	*Resources*
BERT-base (2018)	109M	250B	1.6e20	64 TPU v2 for 4 days (16 V100 GPU for 33 hrs)
GPT-3 (2020)	175B	300B	3.1e23	*~1,000x BERT-base*
PaLM (2022)	540B	780B	2.5e24	*6k TPU v4 for 2 months*

Source

Emergence



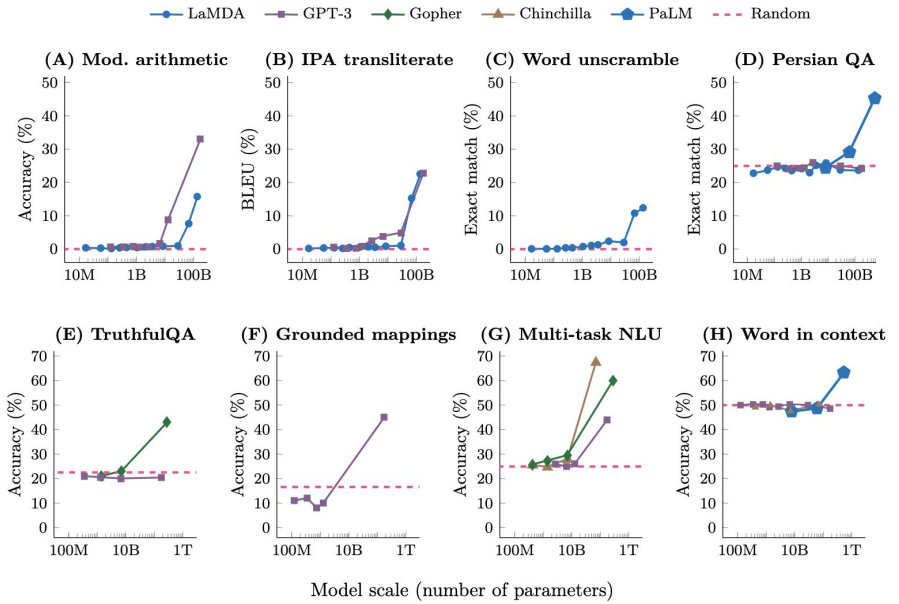


Figure 11: Eight examples of emergence in the few-shot prompting setting. Each point is a separate model.

Source

Source

However..

Q1. When was Justin Bieber born?”
(1994)

Q2. “Who was the champion of the Master’s tournament in 1994?”

Compositional 2 hop question:

Who was the champion of the Master’s Tournament in the year that Justin Bieber was born?

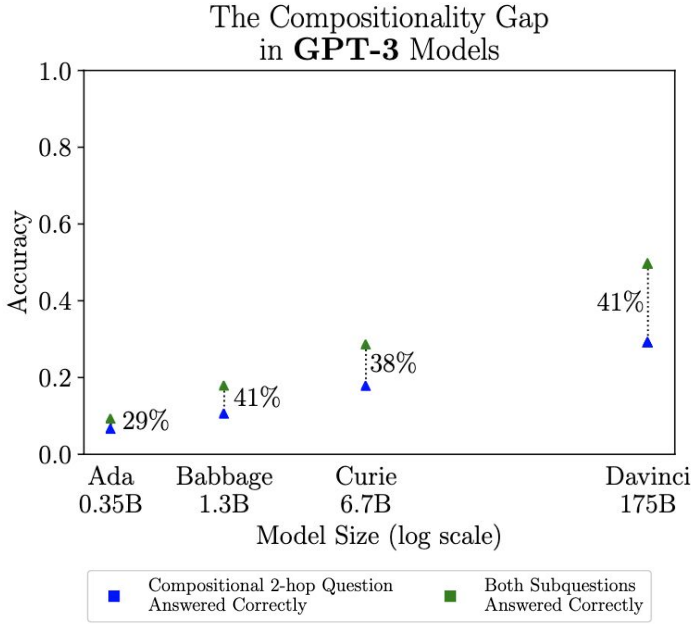
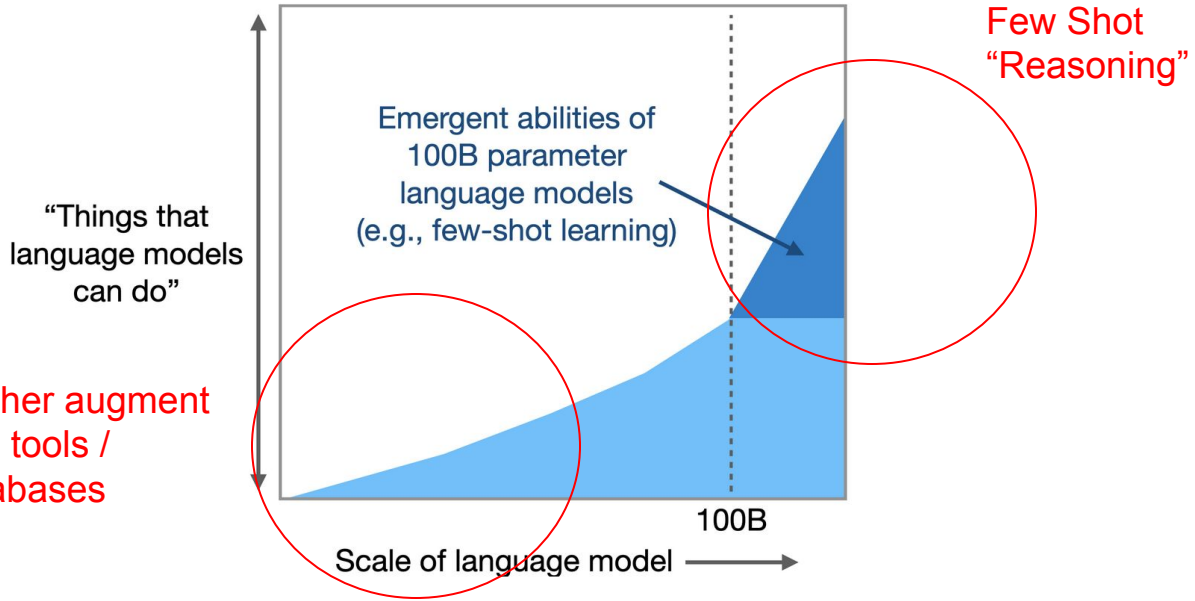


Figure 1: The compositionality gap does not decrease with scale. This plot shows accuracy for the compositional questions (blue) and the two sub-questions that comprise them (green) in the Compositional Celebrities dataset. △ are the regular GPT models and ○ and × are the 001 and 002

Source

Spectrum of ‘augmentation’

My Take



Smaller Language Models

Larger Language Models**

Tool Usage
Retrieval Augmented

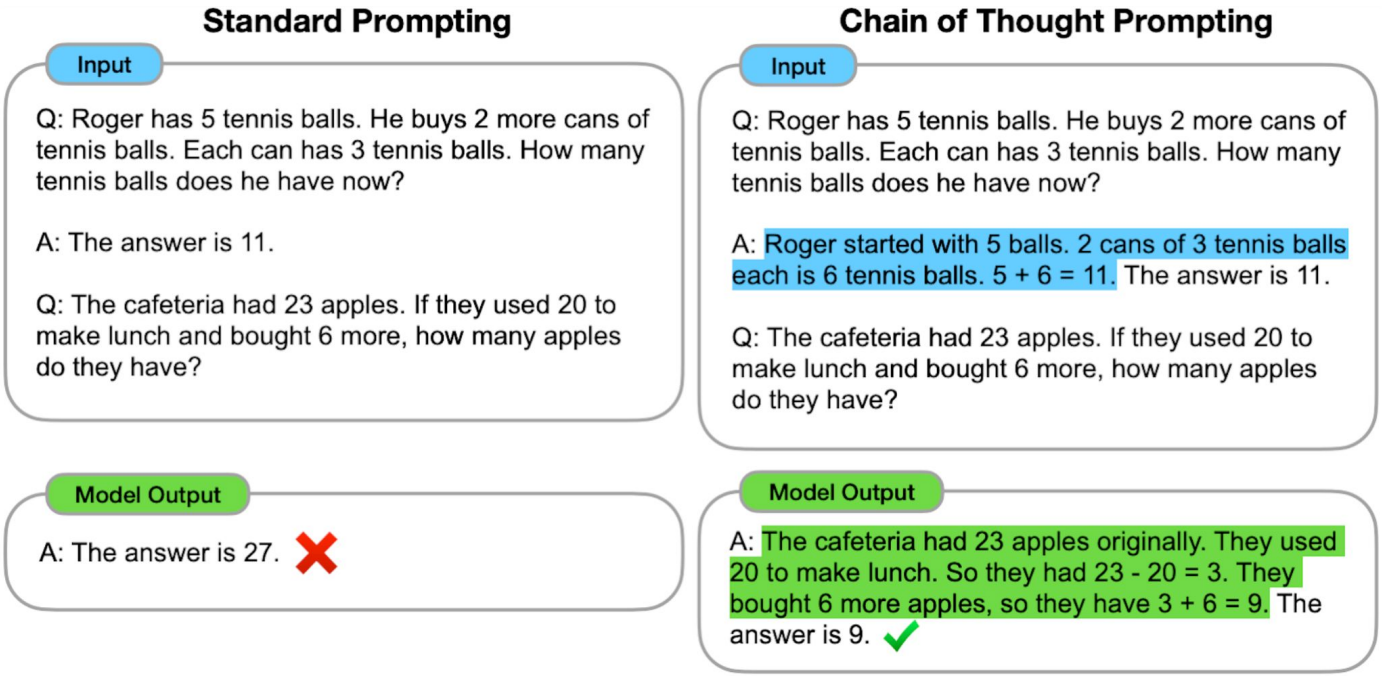
Reasoning

** *some exceptions like UL2*

Reasoning

Chain of Thought (CoT) Prompting

Few Shot Demonstrations



Source

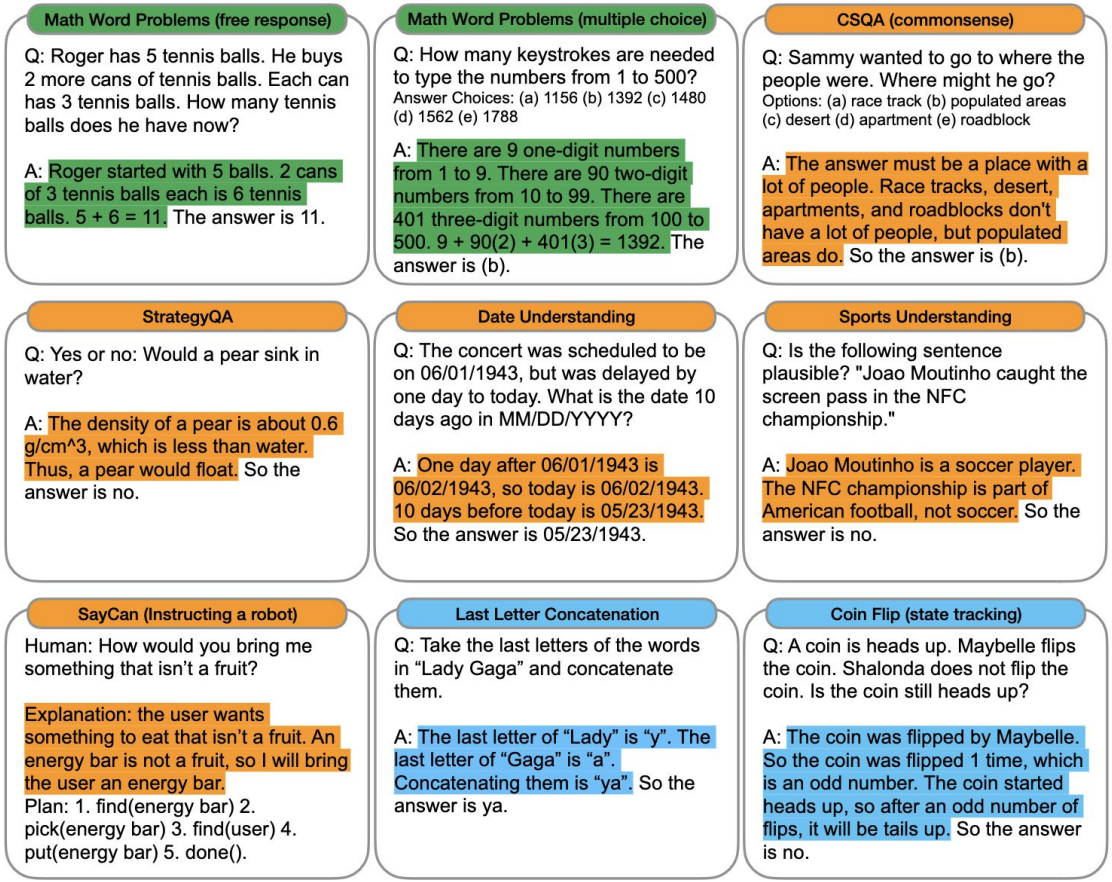
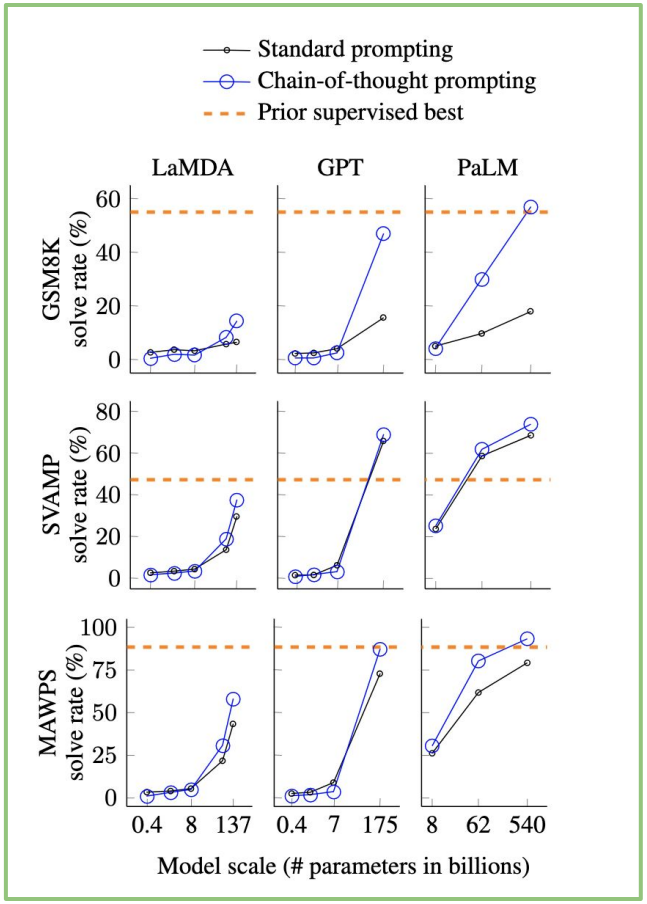


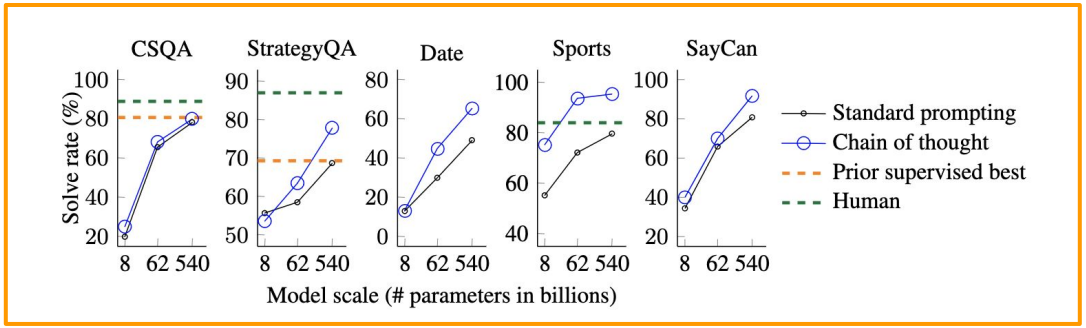
Figure 3: Examples of ⟨input, chain of thought, output⟩ triples for arithmetic, commonsense, and symbolic reasoning benchmarks. Chains of thought are highlighted. Full prompts in Appendix G.

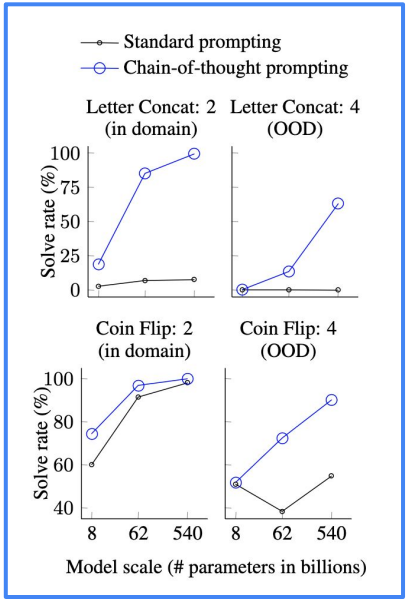
+ Arithmetic Reasoning

+ Common Sense Reasoning

+ Symbolic Reasoning







Benefits of chain of thought prompting only materialize with a sufficient number of model parameters (around 100B).

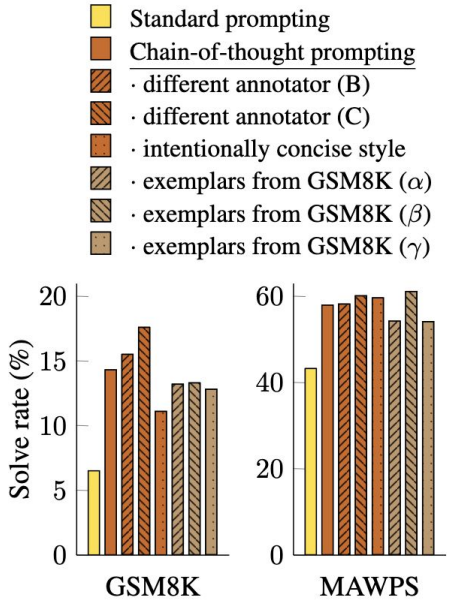


Figure 6: Chain-of-thought prompting has variance for different prompt examples (as expected) but outperforms standard prompting for various annotators as well as for different exemplars.

Robustness

robustness to

- Annotators
- independently-written chains of thought
- different exemplars

Error Analysis on Smaller Models

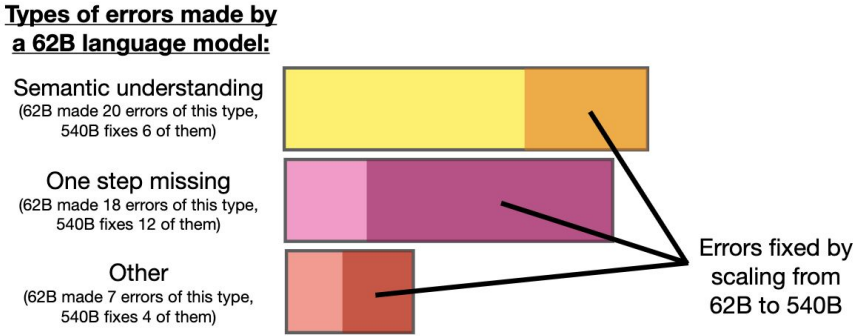


Figure 9: Error analysis of 45 problems that PaLM 62B got incorrect. These errors were categorized that semantic understanding, one step missing, and other. The other category includes hallucinations, repetitive outputs, and symbol mapping errors. Scaling PaLM to 540B fixed a substantial portion of errors in all categories.

Small models:

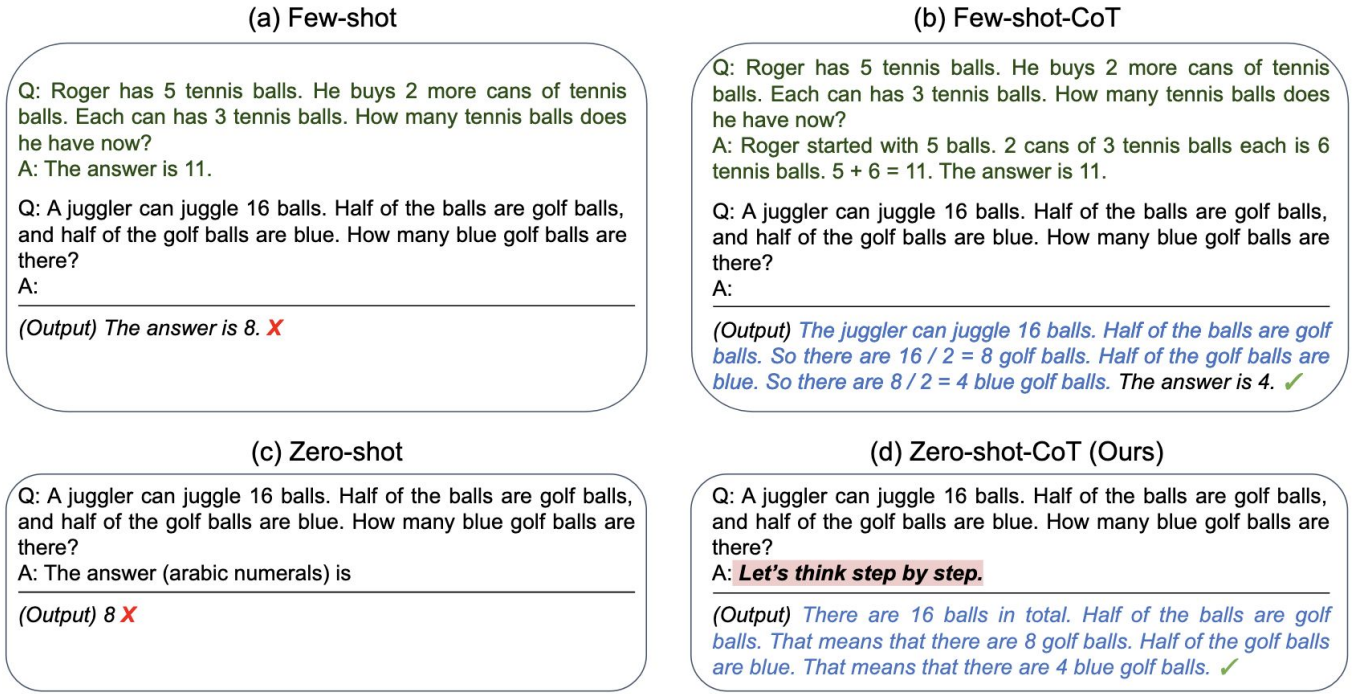
1. *Even simple operations need larger scale*
2. *Did not generate an answer that could even be parsed*

Chain of Thought (CoT) Prompting

Zero Shot
Demonstrations

Chain of Thought (CoT) Prompting

Zero Shot Demonstrations



Source

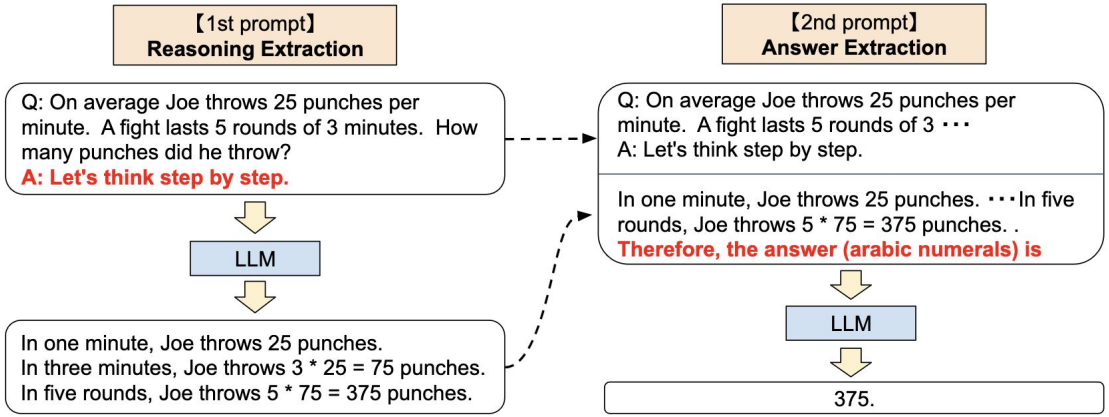
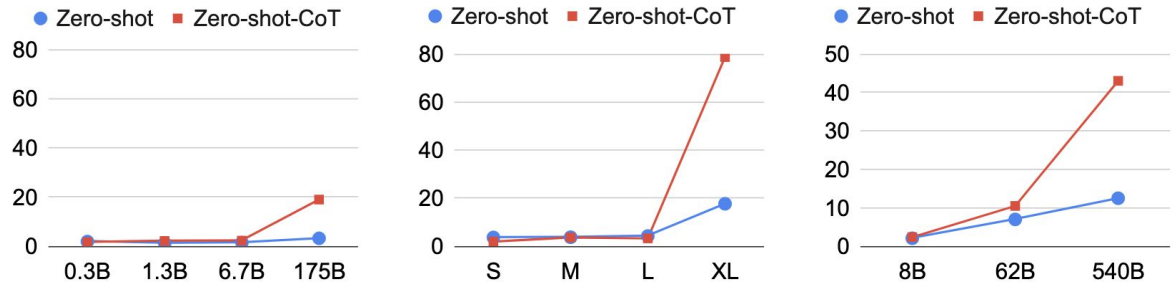


Figure 2: Full pipeline of Zero-shot-CoT as described in § 3: we first use the first "reasoning" prompt to extract a full reasoning path from a language model, and then use the second "answer" prompt to extract the answer in the correct format from the reasoning text.



(a) MultiArith on Original GPT-3 (b) MultiArith on Instruct GPT-3 (c) GMS8K on PaLM

Table 23: Categorization results of produced chain of thought for MultiArith datasets. (*1) These categories are cited from Wei et al. [2022].

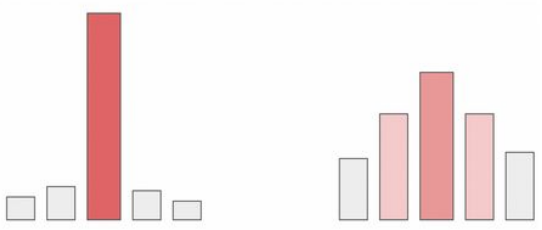
Prediction	CoT Category	Zero-Shot-CoT (%)	Few-Shot-CoT (%)
Correct	CoT is correct	94.0	98.0
	CoT is incorrect	6.0	2.0
Incorrect	CommonSense Mistake	10.0	**23.8**
	Factual Mistake	2.0	0.0
	Logical Mistake	68.0	73.8
	- Calculator error (*1)	(8.)	(**26.2**)
	- Symbol mapping error (*1)	(4.)	(2.4)
	- One step missing error (*1)	(6.)	(7.1)
	- One unnecessary step error	(**10.**)	(2.4)
	- More complicated	(40.)	(35.7)
	Others	**20.0**	2.4

Interesting difference in nature of errors

Table 4: Robustness study against template measured on the MultiArith dataset with text-davinci-002. (*1) This template is used in Ahn et al. [2022] where a language model is prompted to generate step-by-step actions given a high-level instruction for controlling robotic actions. (*2) This template is used in Reynolds and McDonell [2021] but is not quantitatively evaluated.

No.	Category	Template	Accuracy
1	instructive	Let's think step by step.	**78.7**
2		First, (*1)	77.3
3		Let's think about this logically.	74.5
4		Let's solve this problem by splitting it into steps. (*2)	72.2
5		Let's be realistic and think step by step.	70.8
6		Let's think like a detective step by step.	70.3
7		Let's think	57.5
8		Before we dive into the answer,	55.7
9		The answer is after the proof.	45.7
10	misleading	Don't think. Just feel.	18.8
11		Let's think step by step but reach an incorrect answer.	18.7
12		Let's count the number of "a" in the question.	16.7
13		By using the fact that the earth is round,	9.3
14	irrelevant	By the way, I found a good restaurant nearby.	17.5
15		Abrakadabra!	15.5
16		It's a beautiful day.	13.1
-		(Zero-shot)	17.7

CoT self consistency



[Image Source]

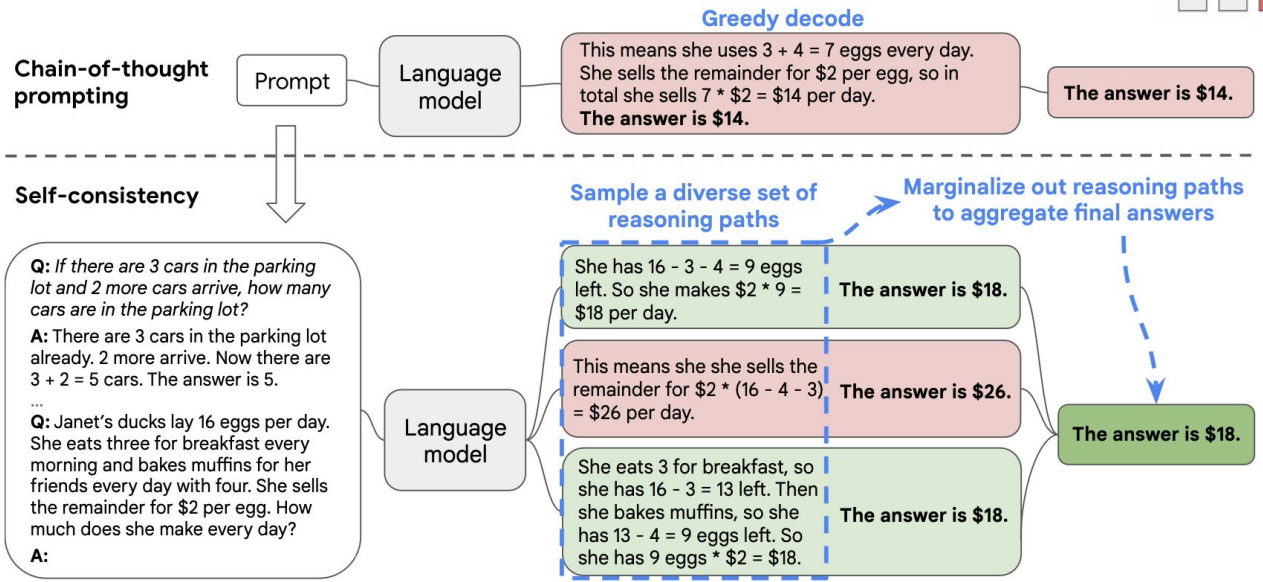


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

[Source]

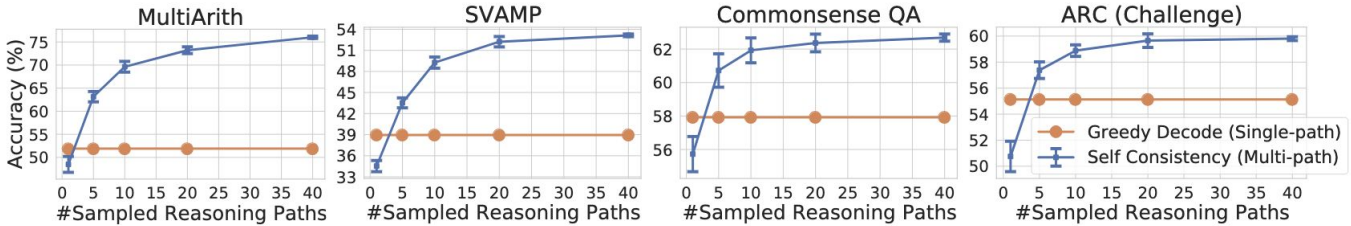


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

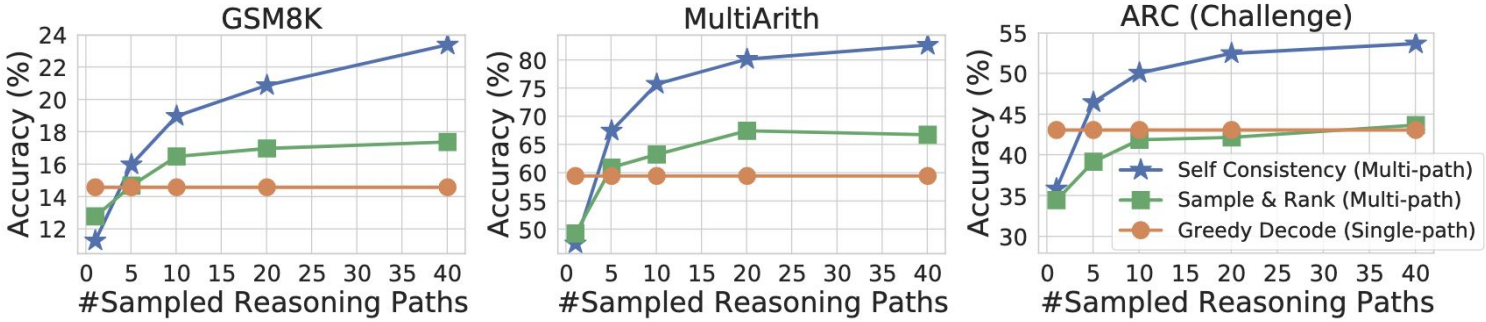


Figure 3: Self-consistency significantly outperforms sample-and-rank with the same # of samples.

Observe the trend of increase. Any comments?

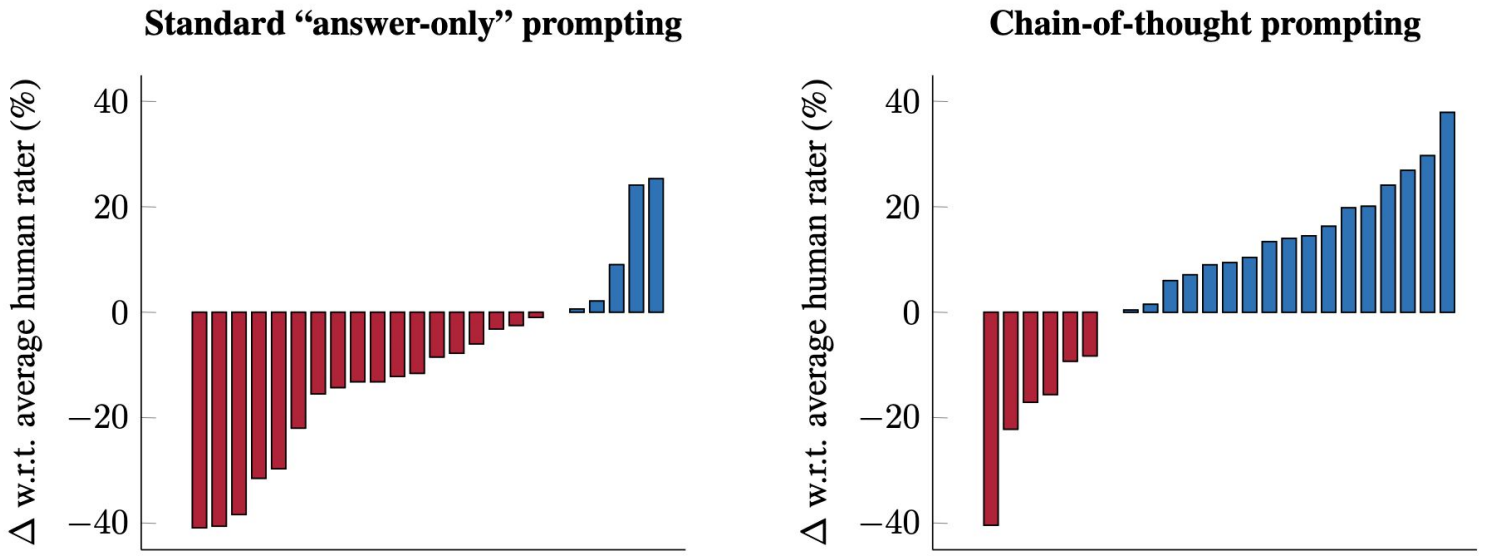
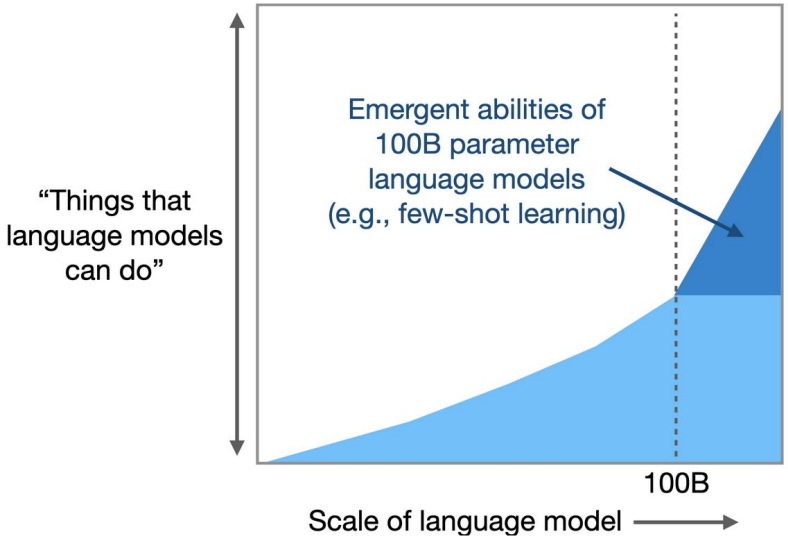
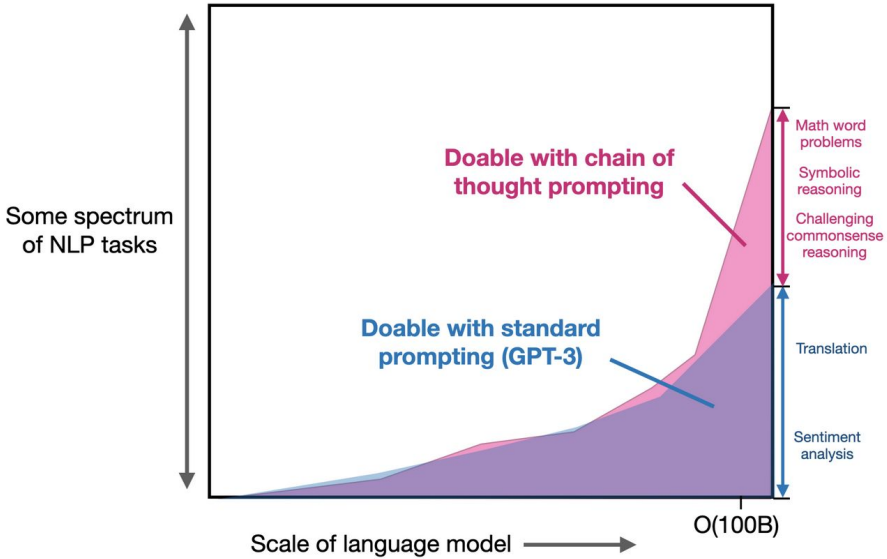


Figure 1: Per-task delta between Codex (code-davinci-002) and the average human-rater performance on 23 challenging tasks in BIG-Bench Hard, for standard *“answer-only”* (left) and *chain-of-thought* (right) prompting.

Source

Closing the compositionality gap?



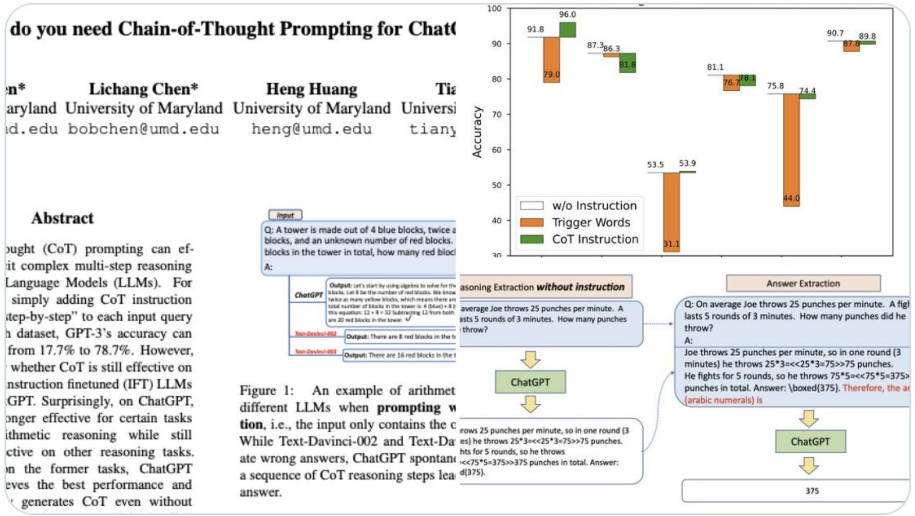


Source

However,

CoT may not always be effective for ChatGPT , depends on the task

#ChatGPT spontaneously follows an instruction (#CoT) even when no instruction is given. But its performance drops when explicitly provided with #CoT. This could be a result of instruction memorization/overfitting during instruction finetuning (IFT). Paper: arxiv.org/pdf/2304.03262...



2 5 19 2,131

Source

“Recursive”

Least to most prompting

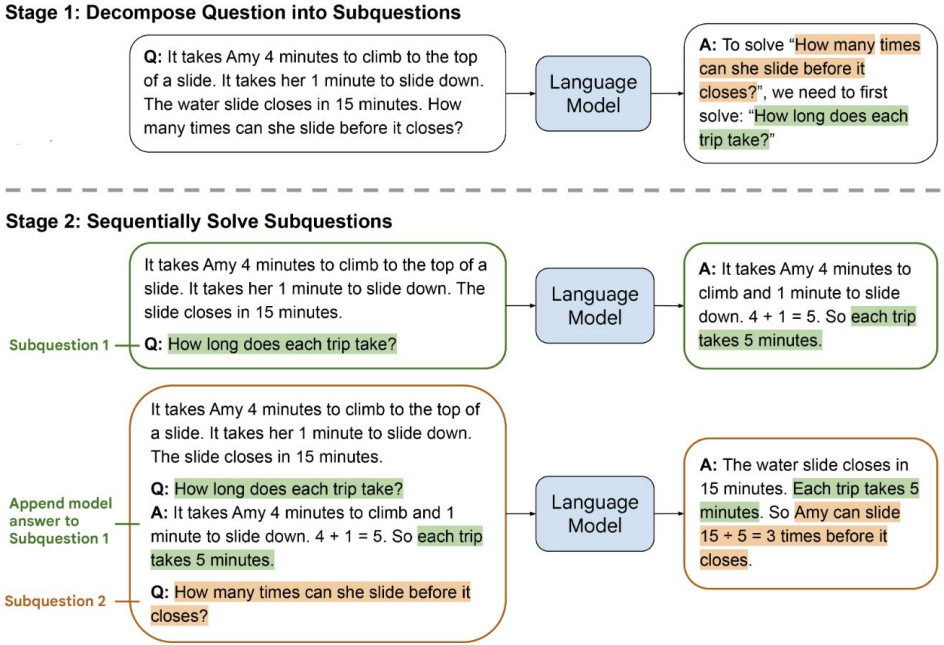
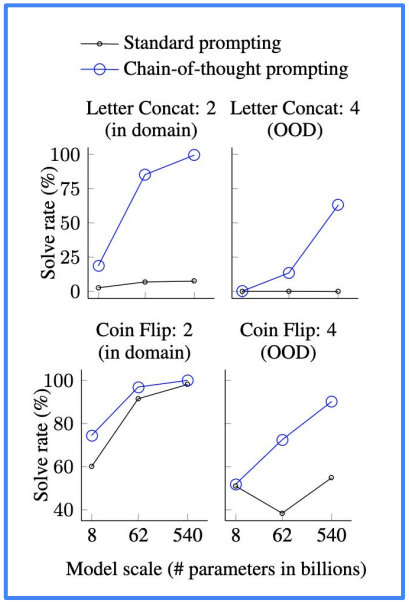


Figure 1: Least-to-most prompting solving a math word problem in two stages: (1) query the language model to decompose the problem into subproblems; (2) query the language model to sequentially solve the subproblems. The answer to the second subproblem is built on the answer to the first subproblem. The demonstration examples for each stage’s prompt are omitted in this illustration.

Source

	$L = 4$	$L = 6$	$L = 8$	$L = 10$	$L = 12$
Standard prompting	0.0	0.0	0.0	0.0	0.0
Chain-of-Thought	84.2	69.2	50.2	39.8	31.8
Least-to-Most	**94.0**	**88.4**	**83.0**	**76.4**	**74.0**

Table 4: Accuracies of different prompting methods on the last-letter-concatenation task. The length of testing lists increases from 4 to 12.



Minerva

Data source	Proportion of data	Tokens	Present during pretraining
Math Web Pages	47.5%	17.5B	No
arXiv	47.5%	21.0B	No
General Natural Language Data	5%	>100B	Yes

PaLM + 118GB dataset of scientific papers using latex, arxiv

Question: A line parallel to $y = 4x + 6$ passes through $(5, 10)$. What is the y-coordinate of the point where this line crosses the y-axis?

Target: Since the line is parallel to $y = 4x + 6$, we know that its slope is 4. Thus the equation of the line is $y = 4x + b$ where b is the y-intercept. Plugging $(5, 10)$ into this gives $10 = 4 \cdot 5 + b \Rightarrow b = \boxed{-10}$ which is what we wanted.

Model output: The line is parallel to $y = 4x + 6$, which means that it has the same slope as $y = 4x + 6$. Since the line passes through $(5, 10)$, we can use the point-slope form of a line to find the equation of the line:

$$y - 10 = 4(x - 5)$$
$$y - 10 = 4x - 20$$
$$y = 4x - 10$$

Now that we have the equation of the line, we can find the y-coordinate of the point where the line crosses the y-axis by substituting $x = 0$ into the equation:

$$y = 4 \cdot 0 - 10 = \boxed{-10}.$$

Figure 1: Question from the MATH dataset (Algebra 4) with a solution generated by Minerva 62B. The solution was selected by majority vote over 256 samples. This and other MATH questions were prompted with the same 4-shot prompt. The model derives the solution by writing down a line equation, simplifying it, substituting x, and solving for y. The model outputs valid LaTeX along with natural language explanations. The approach taken by the model is different from the one used in the ground truth target.

Source

Galactica

- step-by-step reasoning is wrapped with a special token <work> and </work> to mimic an internal working memory.
- At inference time, the model can be asked explicitly to activate this reasoning mode via the <work> token.

Question: A needle 35 mm long rests on a water surface at 20 ◦ C. What force over and above the needle's weight is required to lift the needle from contact with the water surface? $\sigma = 0.0728m$.
<work>

$$\sigma = 0.0728N/m$$
$$\sigma = F/L$$
$$0.0728 = F/(2 \times 0.035)$$
$$F = 0.0728(2 \times 0.035)$$

```
calculate.py
"‘
f = 0.0728*(2*0.035)
with open("output.txt", "w") as file:
file.write(str(round(f, 5)))
"’
```

«run: `calculate.py`»

«read: `output.txt`»

0.0051

</work>

Answer: $F = 0.0051N$

Figure 4: Working memory example from Taylor et al. (2022). This prompt and its output are seen during LM pre-training.

Source

Scratchpad

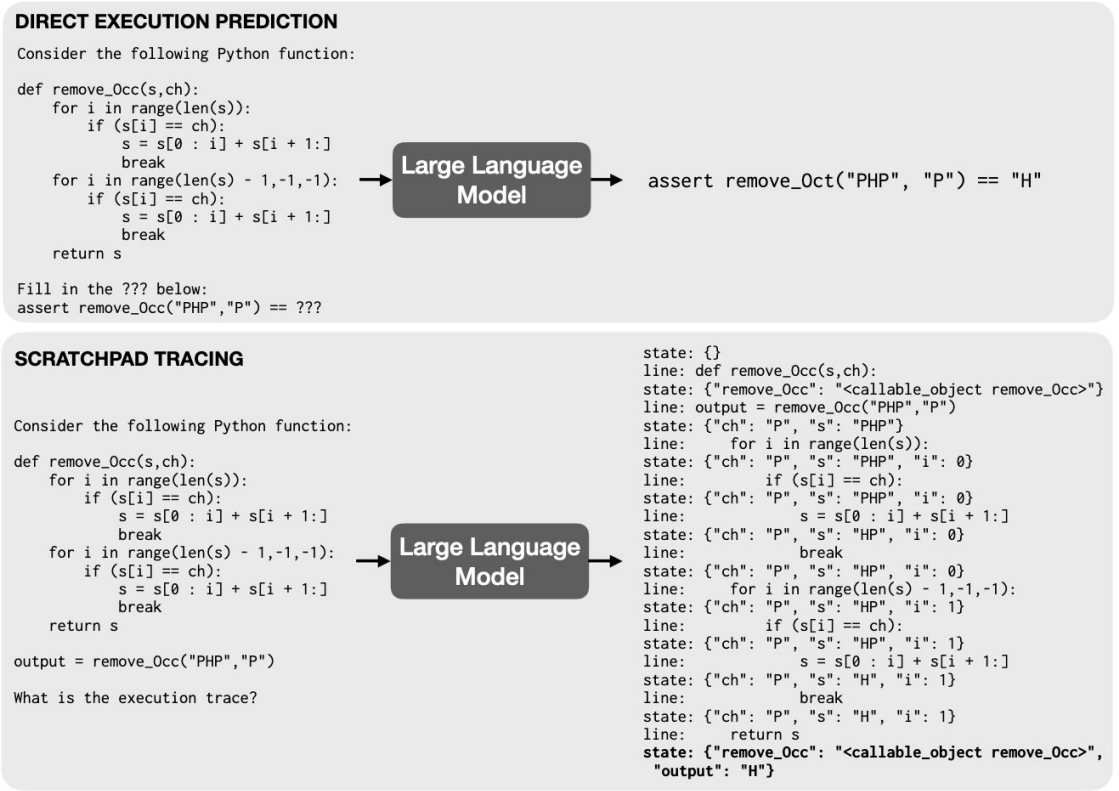


Figure 1: Overview of our scratchpad approach applied to predicting code execution and comparison to direct execution prediction. Top: Previous work has shown that large pre-trained models achieve poor performance when asked to directly predict the result of executing given computer code (Austin et al., 2021). Bottom: In this work, we show that training models to use a *scratchpad* and predict the program execution trace line-by-line can lead to large improvements in execution prediction performance. N.B. Although the example above only has one loop iteration for each loop, all loops are unrolled across time.

Source

Algorithmic Prompting

Table 1: Comparison of different prompting strategies studied in this work. The number of ★ indicates the level to which each strategy exhibits the given characteristic. In this work, we refer to the basic approach of presenting only input-target pairs with no additional explanation as *few-shot*, and we refer to prompts that provide explicit instructions but no running examples of a task as *instruction-only*. We see that algorithmic prompt includes both qualities of natural language explanation and explicit intermediate computations.

Prompt strategy	**Input-target pairs**	**Natural language rationale**	**Intermediate computations**	**Rationale diversity**
Few-shot	★★★	-	-	-
Chain-of-thought	★★★	★★★	★	★★★
Scratchpad	★★★	-	★★	-
Instruction-only	-	★★★	-	★★★
Algorithmic	★★★	★★	★★★	★

Source

A.3 Additional results on two-number addition

This section includes additional details and results for Section 3. In Figure 10, we provide an illustration of different prompting strategies for two-number addition with differing levels of detail in the explanation.

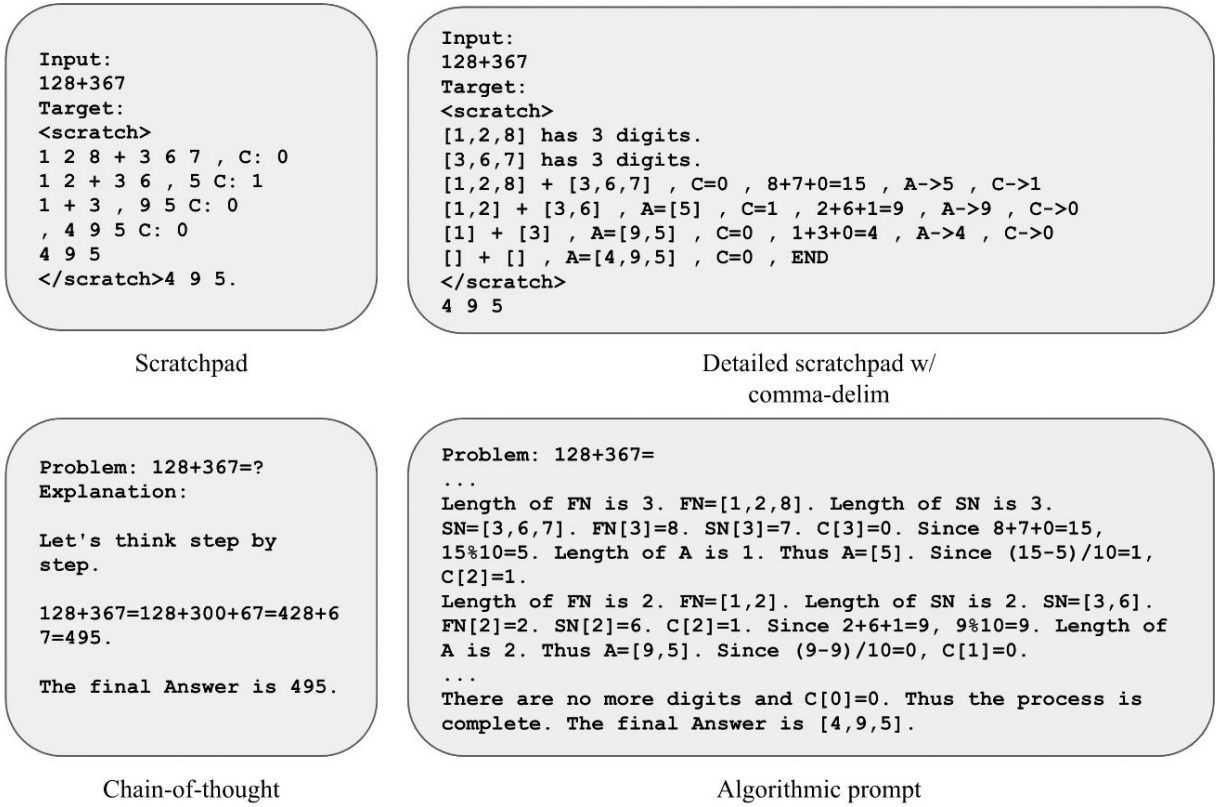


Figure 10: Examples of the two-number addition prompt using different techniques.

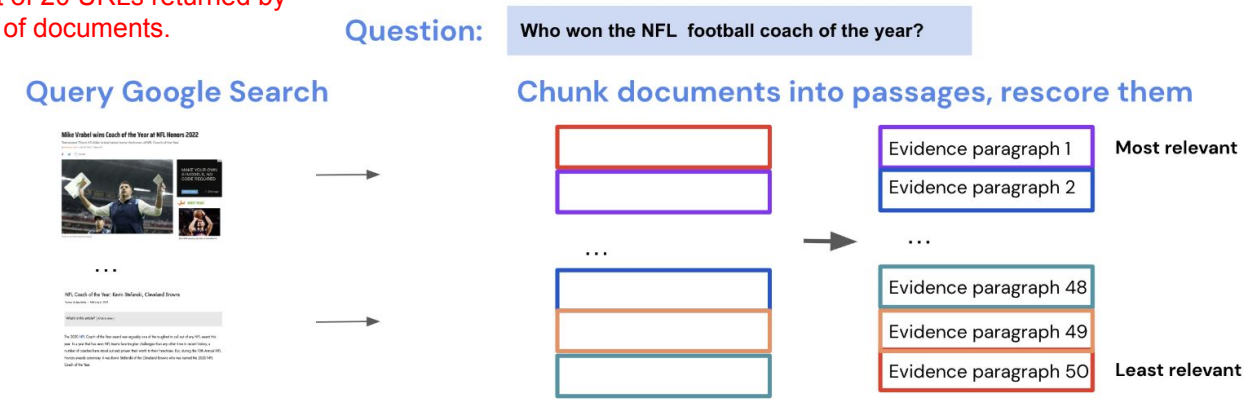
Reasoning

Access Tools

Internet Augmented LMs

Split into paragraphs of 6 sentences
Rank based on TF-IDF based cosine similarity between paragraphs and question

Given a question,
clean text is extracted out of 20 URLs returned by Google, resulting in a set of documents.



Source

Internet Augmented LMs

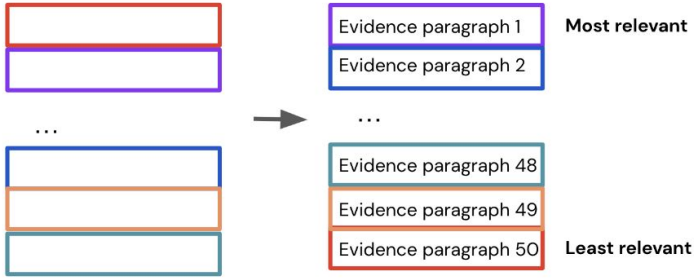
Split into paragraphs of 6 sentences
Rank based on TF-IDF based cosine similarity between paragraphs and question

Given a question,
clean text is extracted out of 20 URLs returned by Google, resulting in a set of documents.

Question: Who won the NFL football coach of the year?

Query Google Search

Chunk documents into passages, rescore them



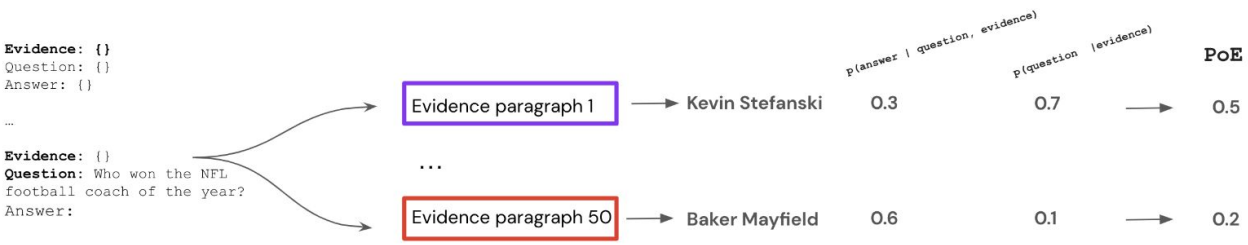
Candidate answers for each paragraph

k-shot prompt

Generate answers using LM

Generate scores

Rerank



Combine both: PoE

"Noisy Channel", maximise:

$$p(a_i \mid q) = \frac{p_{\text{LM}}(q|a_i,p_i)\cdot p_{\text{LM}}(a_i|p_i)}{p_{\text{LM}}(q|p_i)}$$

"RAG"

$$p(a_i \mid q) = \sum_{i=1}^{n} p_{\text{tf-idf}}(p_i \mid q) \cdot p_{\text{LM}}(a_i \mid q, p_i)$$

Source

Dataset	SOTA	#sampled answers: 1		#sampled answers: 200			
		CB	$OB^{\text{no reranking}}_{\text{Google}}$	CB	OB_{Gold}	$OB^{\text{a\|q,p}}_{\text{Google}}$	$OB^{\text{PoE}}_{\text{Google}}$
NQ	51.4 [8]	21.7	23.1	25.8	61.7	32.7	38.4
HotpotQA	65.2 [28]	20.7	24.5	21.2	54.8	26.3	30.3
Fever	73.2 [31]	44.5	52.2	44.5	66.6	52.0	57.2
StrategyQA	63.6 [29]	61.0	61.1	61.0	80.4	64.6	66.2

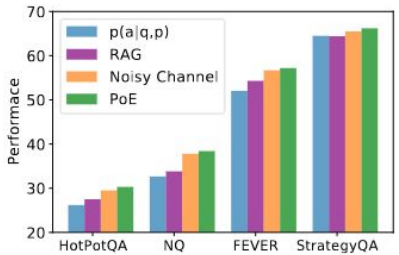
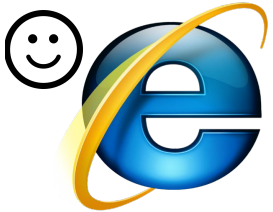


Figure 1: Question answering performance on all 4 datasets using Gopher-280B OB_{Google} model in combination with different answer scoring functions for reranking the answers generated for each of the top 50 Google retrieved paragraphs.

Scaling via inference time, not number of parameters or train time compute

“searching the Internet is worth more than 273 billion parameters.”



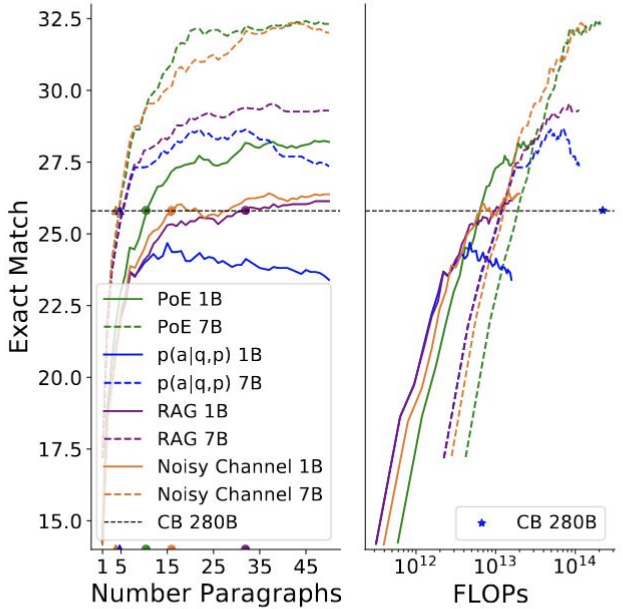


Figure 3: Scaling analysis on the NQ dataset: exact match as a function of number of conditioned paragraphs (left) and FLOPs (right).

Scaling via inference time, not number of parameters or train time compute

"searching the Internet is worth more than 273 billion parameters."

Natural follow up: learn to search

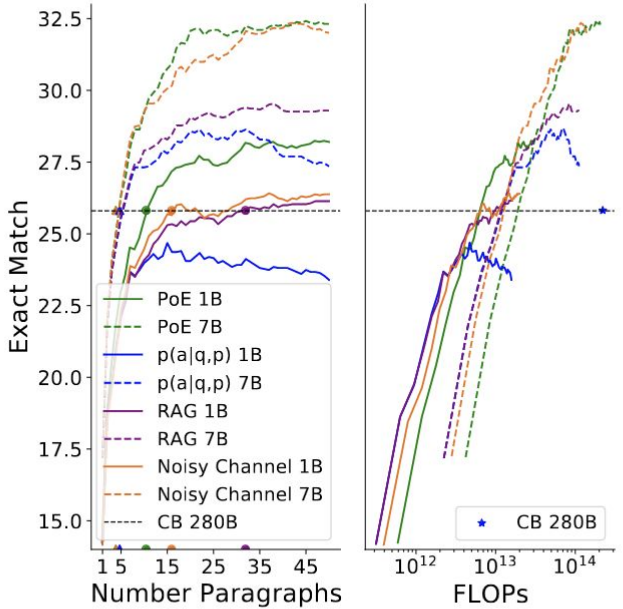


Figure 3: Scaling analysis on the NQ dataset: exact match as a function of number of conditioned paragraphs (left) and FLOPs (right).

SelfAsk

Few shot prompts to demonstrate how / what to ask and then let SelfAsk keep asking / decide when to stop

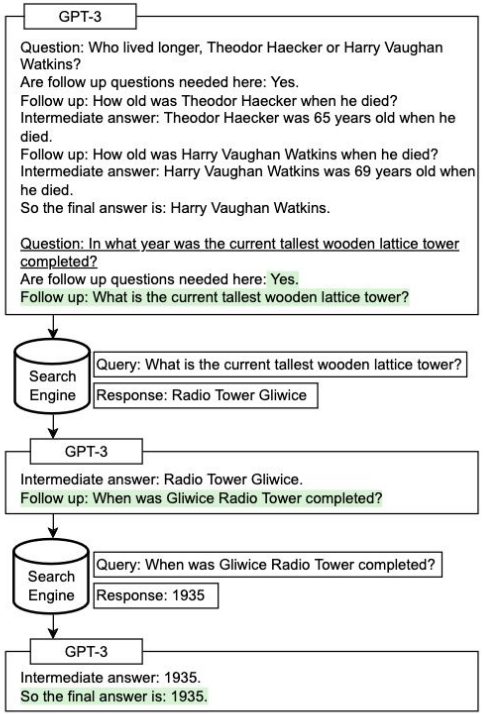


Figure 5: Self-ask + Search Engine: prompt on white background, LM-generated text in green. We start by using a few-shot prompt (reduced here for space) and appending the test-time question (underlined) to it. The LM then generates a follow-up question which we input to an internet search engine. The response is inserted back into the rest of the prompt to let the LM generate the next follow-up question. This process then repeats until the LM decides to output the final answer.

Source

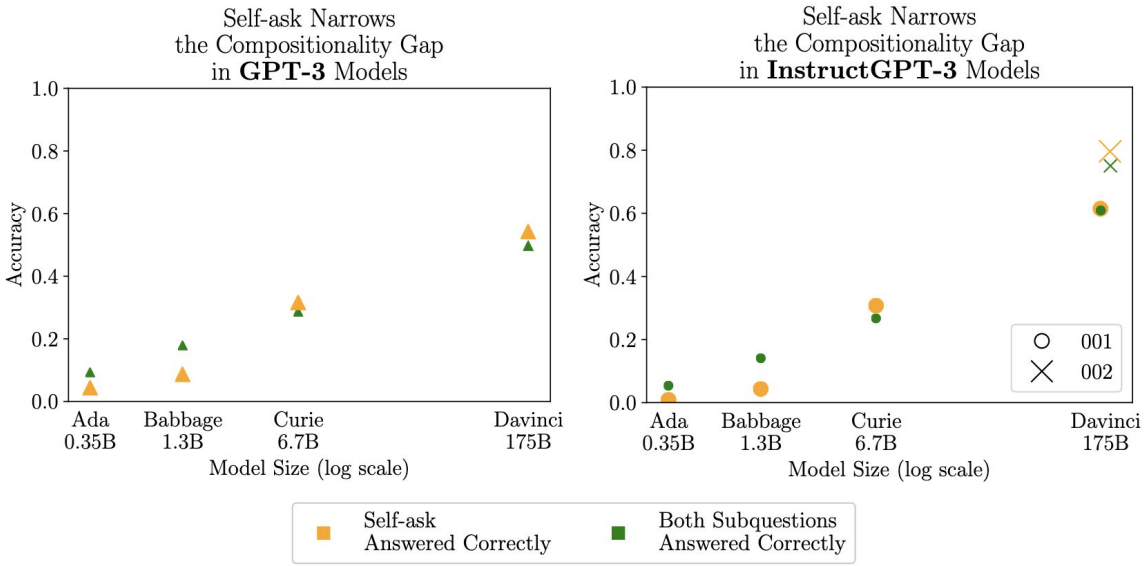


Figure 4: Self-ask is able to narrow and sometimes close the compositionality gap on CC. Here, self-ask uses a 1-shot prompt. Chain of thought performed within 1% of self-ask on this dataset and is not shown here. △ are the regular GPT models and ○ and × are the 001 and 002 instruct models.

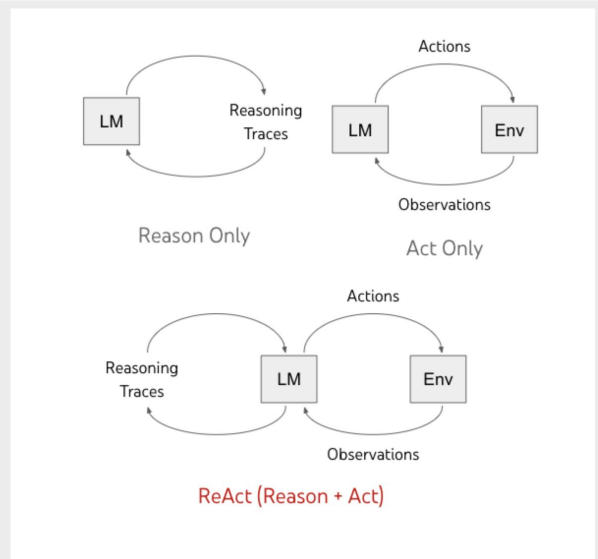
ReACT

Synergizing Reasoning and Acting in Language Models

Frozen large language model, PaLM-540B is prompted with few-shot in-context
examples to generate

- domain-specific actions and
- free-form language thoughts for task solving

- Each in-context example is a human trajectory of actions, thoughts, and environment observations to solve a task instance



Previous methods prompt language models (LM) to either generate self-conditioned reasoning traces or task-specific actions. We propose ReAct, a new paradigm that combines reasoning and acting advances in language models.

ReAct

Question	What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
Thought 1	I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area.
Action 1	Search[Colorado orogeny]
Observation 1	The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.
Thought 2	It does not mention the eastern sector. So I need to look up eastern sector.
Action 2	Lookup[eastern sector]

Source

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Act 4: Finish[yes] ✗

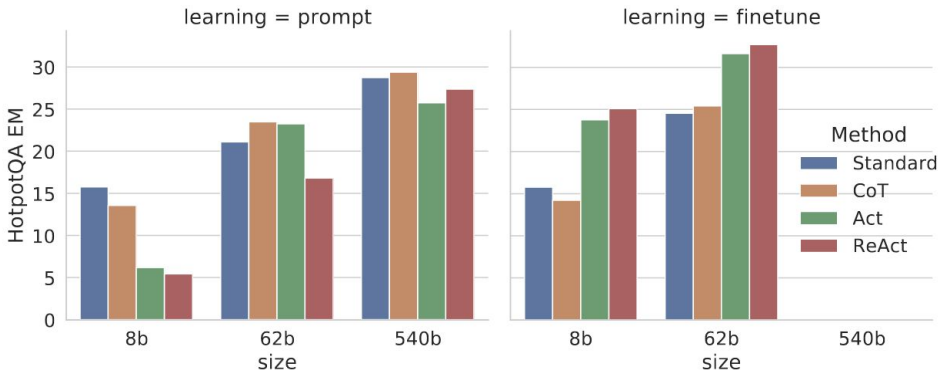
(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software)', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓



PAL

- Main benefit comes from having synergy with interpreter
- <hypothesis> Also trained

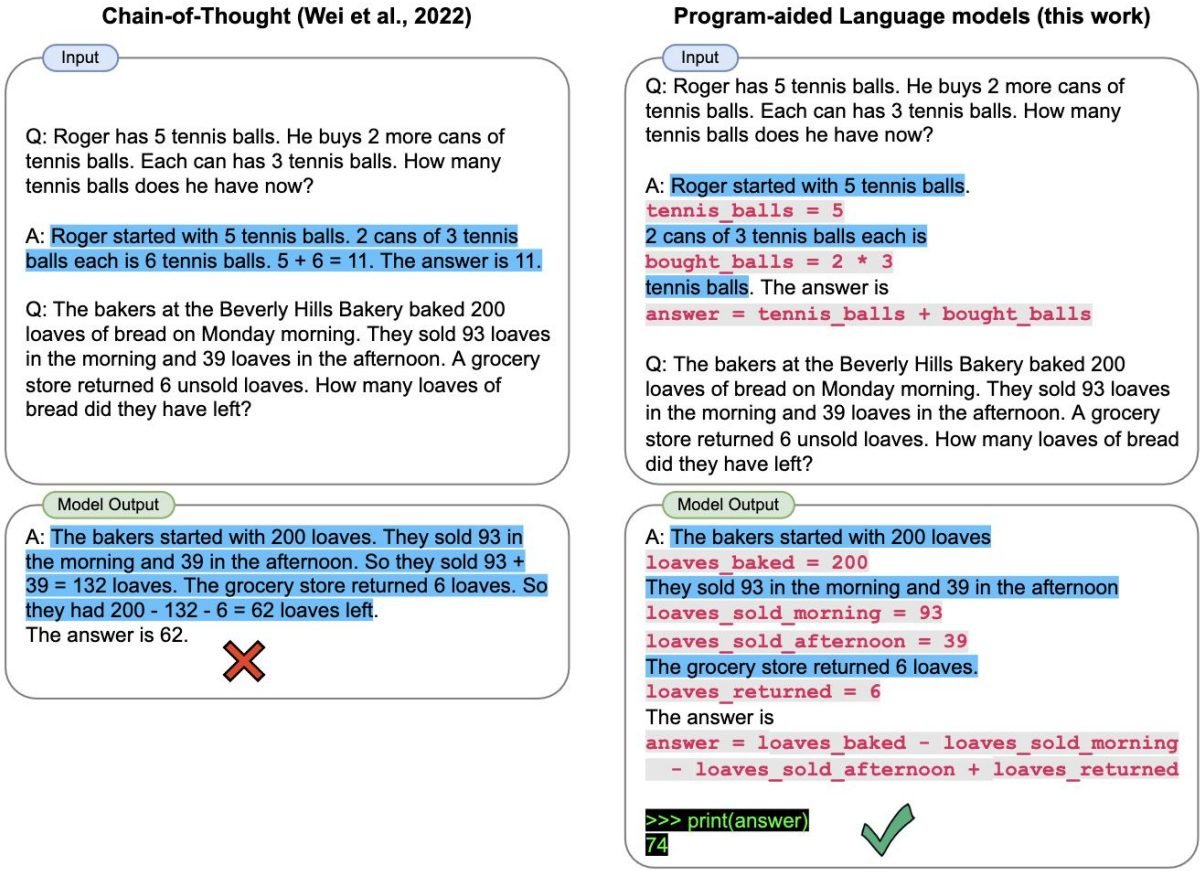


Figure 1: A diagram illustrating PAL: Given a mathematical reasoning question, Chain-of-thought (left) generates intermediate reasoning steps of free-form text. In contrast, Program-aided Language models (PAL, right) generate intermediate steps *and* Python code. This shifts the role of *running* the reasoning steps from the language model to the Python interpreter. The final answer is obtained by running the generated reasoning chain. Chain-of-thought reasoning is highlighted in blue; PAL steps are `highlighted in gray and pink`; the Python interpreter run is highlighted in black and green.

Source

Also POT, not covered

ery in-context example in PAL is a *pair* $\langle x_i,\ t_i \rangle$, where $t_j = [s_1, s_2, \ldots, s_N]$ with each $s_i \in \text{NL} \cup \text{PL}$, a sequence of tokens in either NL or PL. The complete prompt is thus $p \equiv \langle x_1 \cdot t_1 \rangle \parallel \langle x_2 \cdot t_2 \rangle \parallel \ldots \parallel \langle x_k \cdot t_k \rangle$.

Given a test instance x_{test}, we append it to the prompt, and $p \parallel x_{test}$ is fed to the LM. We let the LM generate a prediction t_{test}, which contains both the intermediate steps *and* their corresponding programmatic statements.

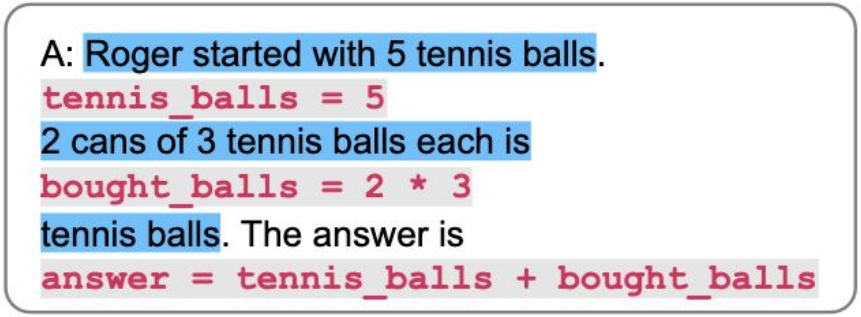


Figure 2: A close-up of a single example from a PAL prompt. Chain-of-thought reasoning is highlighted in blue, and PAL programmatic steps are highlighted in gray and pink.

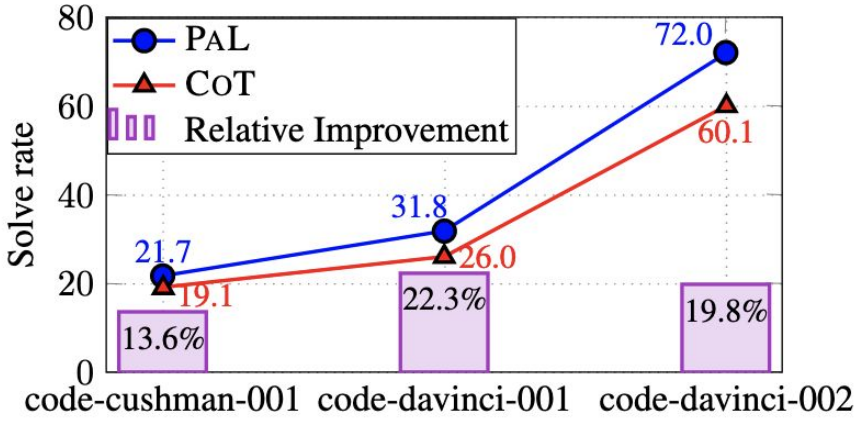


Figure 7: PAL with different models on GSM8K: though the absolute accuracies with `code-cushman-001` and `code-davinci-001` are lower than `code-davinci-002`, the relative improvement of PAL over CoT is consistent across models.

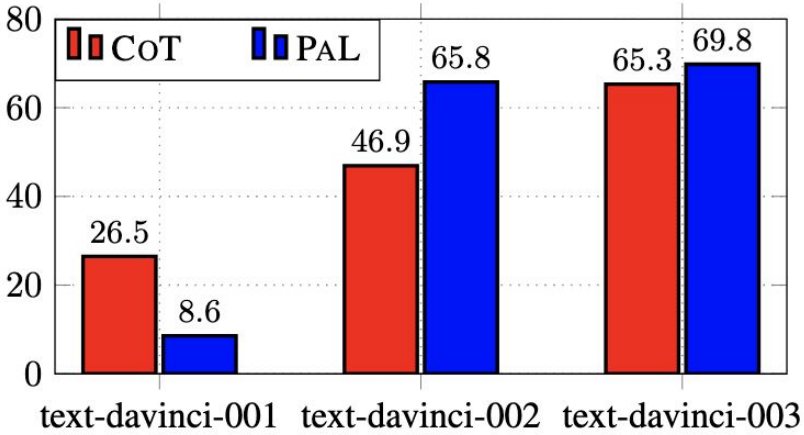


Figure 8: PAL with NL LMs on GSM8K: though CoT outperforms PAL with `text-davinci-001`, once the base LM is sufficiently strong, PAL is beneficial with `text-davinci-002` and `text-davinci-003` as well. That is, PAL is not limited to code-LMs only.

Note: No evaluation on smaller models, all are (100B+)

Clean Code Matters :)

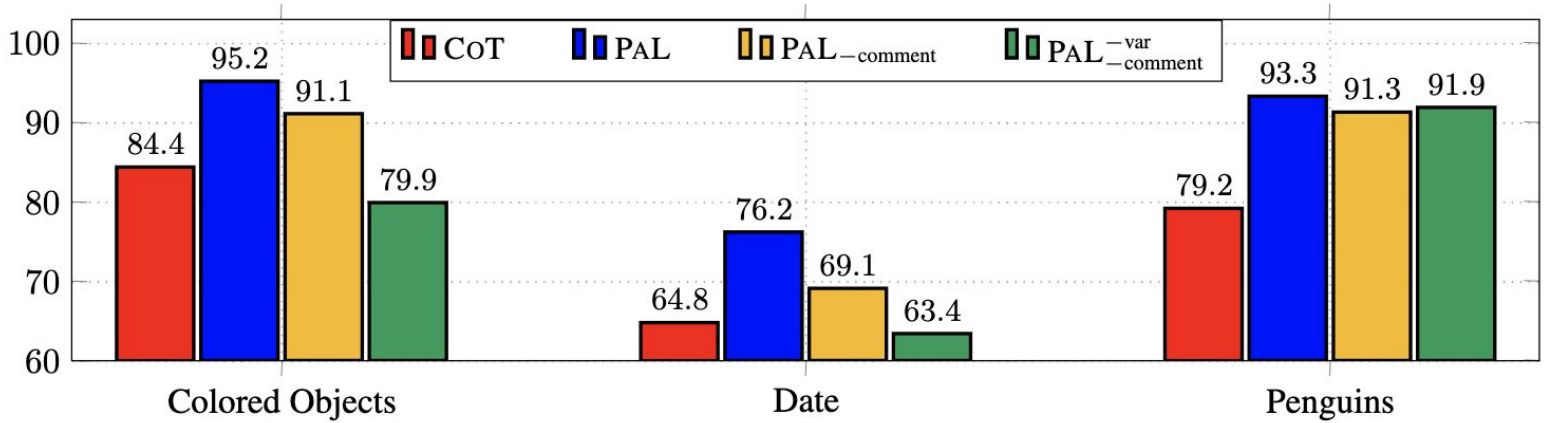


Figure 9: Ablation study of PAL prompt formats. We consider the original PAL prompt, it with natural language comments removed ($\text{PAL}_{-\text{comment}}$), and further variable names replaced with random character ($\text{PAL}^{-\text{var}}_{-\text{comment}}$). As a reference, we also show the CoT performance (blue).

Also works with Least to Most, not only CoT

I. Generalization of PAL to Least-to-Most Prompting

```
Q: Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently
↪  twice 30 years old, how old is Kody?
A: To answer the question "How old is Kody?", we need to know: "How old is Mohamed?",
↪  "How old was Mohamed four years ago?", "How old was Kody four years ago?".
```

(a) Least-to-Most Math Reducing Prompt

```
Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice
↪  30 years old, how old is Kody?

Q: How old was Mohamed four years ago?
A: We were told that Mohamed is currently twice 30 years old, so he is currently 30 *
↪  2 = 60 years old. That means that four years ago he must have been 60 - 4 = 56
↪  years old. The answer is 56.

Q: How old is Kody?
A: Four years ago, Kody was half as old as Mohamed, so Kody must have been 56 / 2 =
↪  28 years old then. Since Kody was 28 years old four years ago, she must now be 28
↪  + 4 = 32 years old. The answer is 32.
```

(b) Least-to-Most Math Solving Prompt

```
# Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice
    30 years old, how old is Kody?

# How old was Mohamed four years ago?
mohamed_age_current = 30 * 2
mohamed_age_4_years_ago = mohamed_age_current - 4

# Final Question: How old is Kody?
kody_age_4_years_ago = mohamed_age_4_years_ago / 2
kody_age_current = kody_age_4_years_ago + 4
answer = kody_age_current
```

(c) PAL Math Solving Prompt

Figure 15: Prompts for Math data sets.

Dataset (500 examples)	LEAST-TO-MOST	LEAST-TO-MOST + PAL
GSM8K	67.2	**72.8**
SVAMP	75.2	**78.2**

Conditioning on code really helps!

Model	Accuracy (%)
OpenAI (`text-davinci-002`)[1]	15.6
OpenAI (`text-davinci-002`) + CoT[1]	46.9
OpenAI (`text-davinci-002`) + CoT + Calculator[1]	46.9
OpenAI (`code-davinci-002`)[1]	19.7
OpenAI (`code-davinci-002`) + CoT[1]	63.1
OpenAI (`code-davinci-002`) + CoT + Calculator[1]	65.4
GPT-3 175B + FT + CoT + Calculator[2]	34.0
GPT-3 175B + FT + CoT + Calculator + Verifier[2]	55.0
PaLM 540B[3]	17.0
PaLM 540B+CoT[3]	54.0
PaLM 540B+CoT+Calculator[3]	58.0
PAL[4]	72.0

Table 1: Evaluation of different reasoning methods on GSM8K, a popular reasoning benchmark. FT denotes fine-tuning and CoT denotes chain-of-thought. The reported accuracies are based on [1]: (Wei et al., 2022c); [2]: (Cobbe et al., 2021); [3]: (Chowdhery et al., 2022); and [4]: (Gao et al., 2022).

Table from survey paper

Other kinds of "code conditioning"

Mindseye

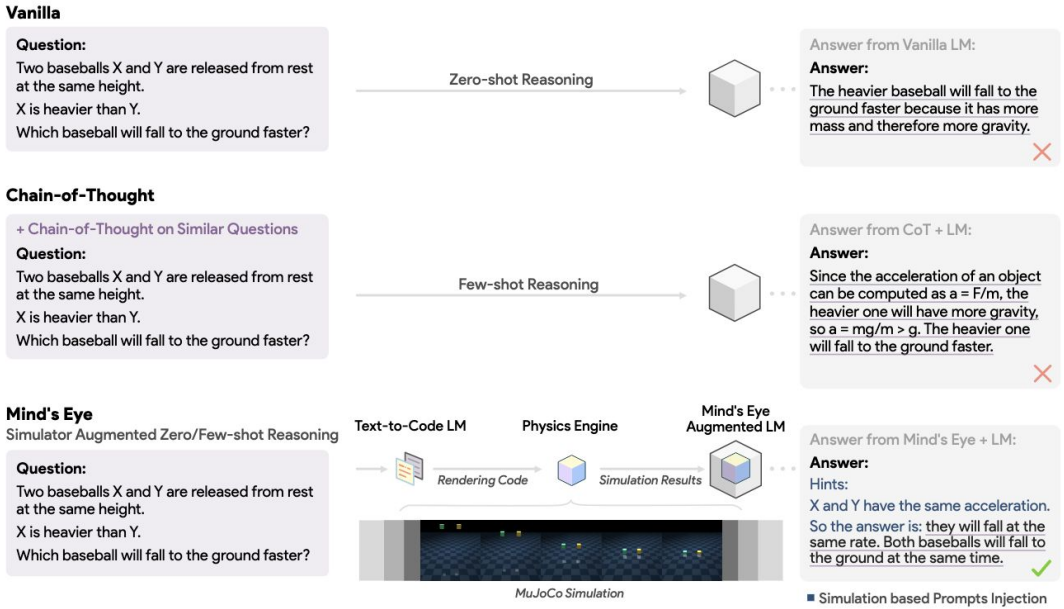
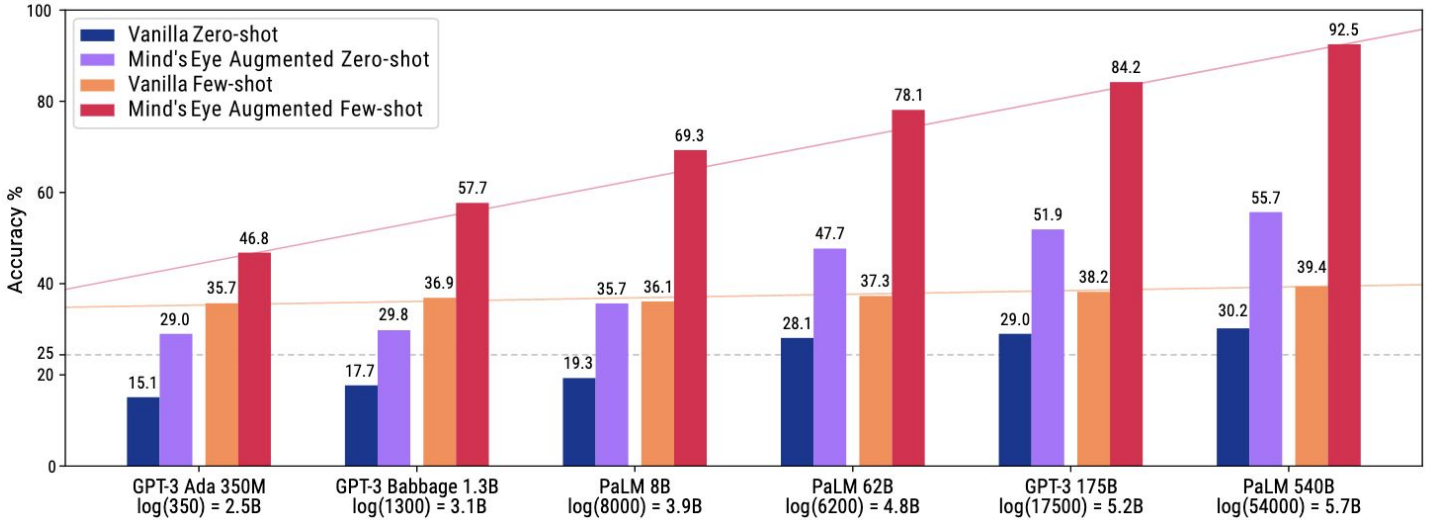


Figure 1: Current language models are still challenged by simple questions that require a good understanding of the physical world. The answer elicited by Chain-of-Thought can still be wrong if the required knowledge is missing or misrepresented in LMs. Mind's Eye, instead, enables grounded LM reasoning by directly simulating the scene in the given question. Then the LM can reason over the injected ground-truth rationale to generate the correct answers.

[Source]

Table 2: Comparison in formulation of how the LM inference process is grounded (given question x generating answer y). With the aid of a simulator (i.e., MuJoCo), Mind's Eye is not only scalable (not requiring human annotation) but also well-grounded with the physical world.

Method Name	Formulation	Scalable?	Grounded?
Zero-shot Reasoner	$y \leftarrow \arg\max_{\hat{y}} \text{LM}(\hat{y}\|x, \textit{“Let's think step by step”})$	✓	✗
Chain-of-Thought	$y \leftarrow \arg\max_{\hat{y}} \text{LM}(\hat{y}\|x, \text{Chain-of-Thought} \sim p_{\text{Human}})$	✗	?
Ours: Mind's Eye	$y \leftarrow \arg\max_{\hat{y}} \text{LM}(\hat{y}\|x, \text{Simulator}(x, \hat{y}) \sim p_{\text{World}})$	✓	✓



toolformer

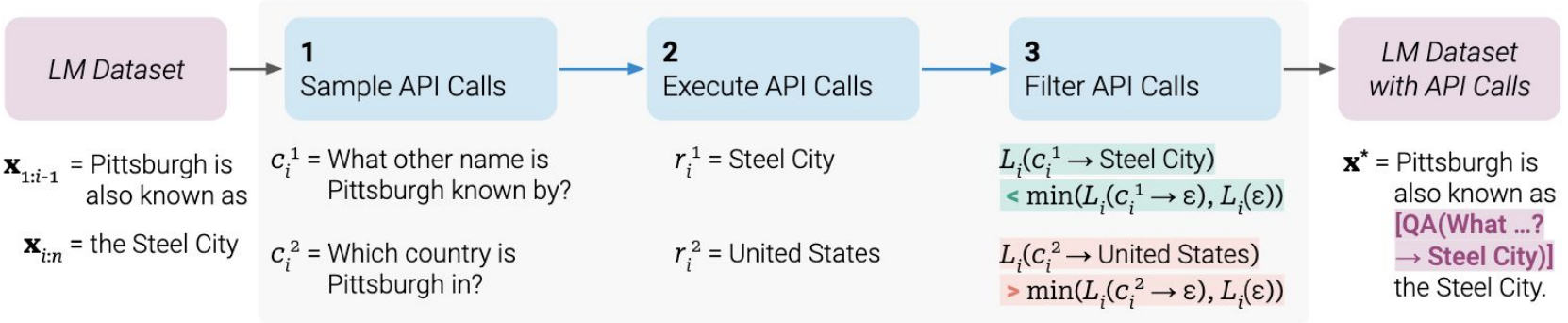


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text $\mathbf{x}$, we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \ldots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text $\mathbf{x}^*$.

Source

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.

Source

1. Demonstrate via few shot learning to inject possible API calls
2. Sample and execute these API calls, keep a top few based on whether they help predict

2. *Filter annotations based on whether API calls help model predict future tokens.* Use a self-supervised loss to decide which API calls are actually helpful.
   - Execute each API call c_i to get corresponding result r_i.
   - Compute weighted cross entropy loss for the LM over tokens $x_i, \ldots, x_n$ when the model is prefixed with the prompt. Two versions are computed, one with API result and the other with empty sequence ε.

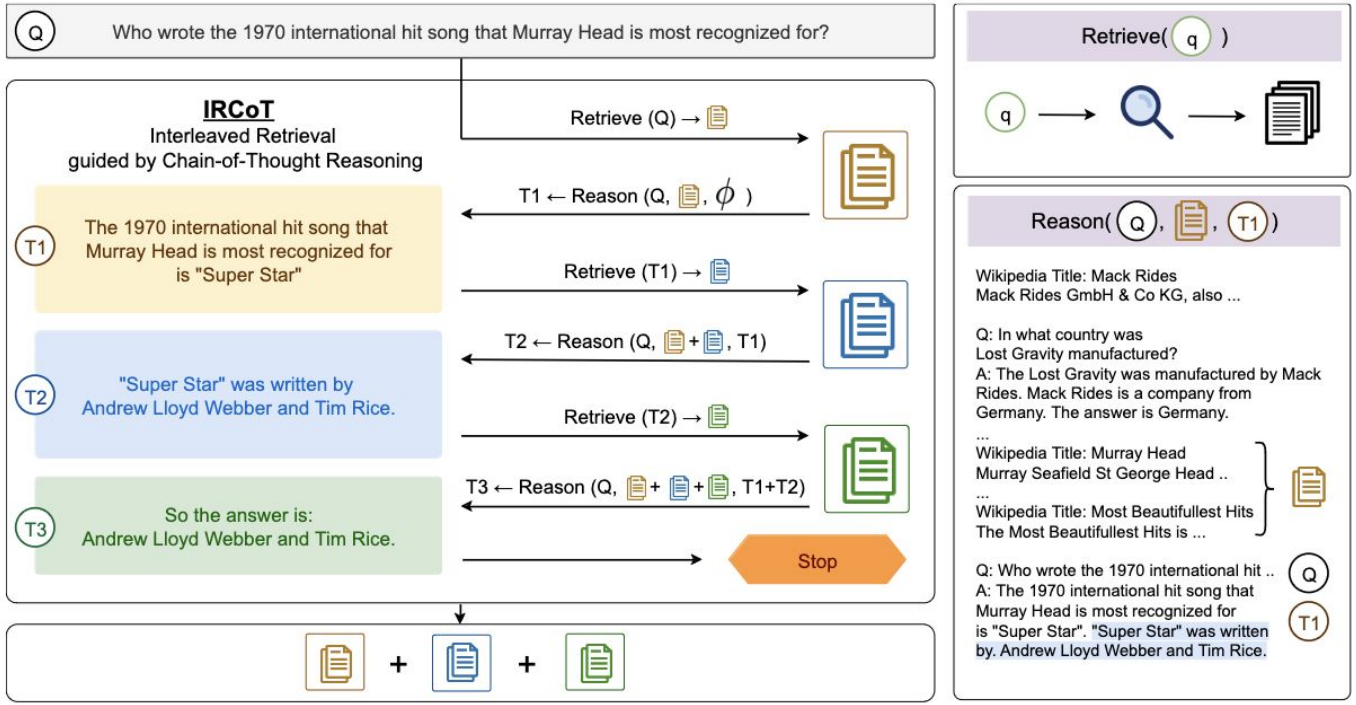
$$L_i^+ = L_i(e(c_i, r_i))$$
$$L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$$

   Only API calls with $L_i^- - L_i^+$ larger than a threshold are kept, meaning that adding this API call and its results help the model predict future tokens.

Source


Reasoning
Access Tools
Retrieval

IRCoT



Generate next CoT sentence using all previous questions and paragraphs

Figure 2: IRCoT interleaves chain-of-thought (CoT) generation and retrieval steps to guide the retrieval by CoT and vice-versa. We start by retrieving K documents using the question as they query and repeat two steps alternatingly until termination. (i) `reason`-step generates next CoT sentence based on the question, so far retrieved paragraphs, and CoT sentences. (ii) `retrieve`-step retrieves K more paragraphs based on the last CoT sentence. The process terminates when the generated CoT has "answer is" or the number of steps exceeds a threshold. The collection of all paragraphs is returned as the retrieval result on the termination.

Source

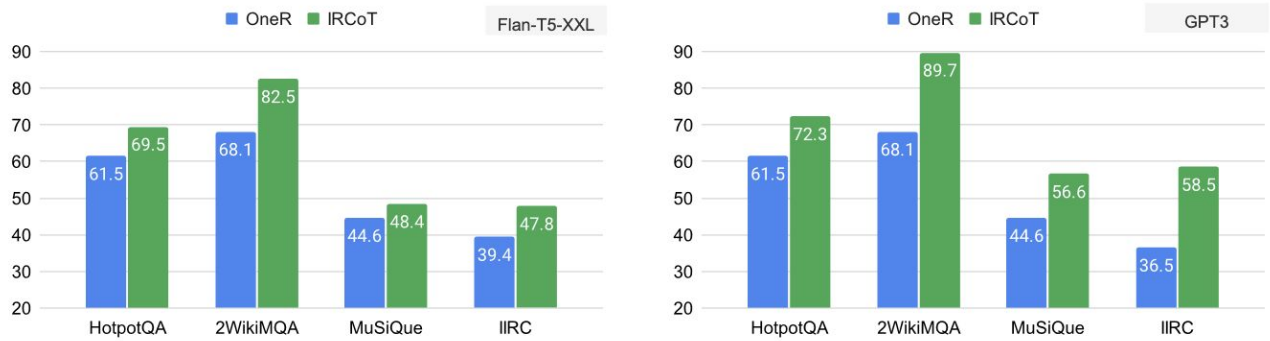


Figure 3: Retrieval recall for one-step retriever (OneR) and IRCoT instantiated from `Flan-T5-XXL` (left) and `GPT3` (right) models. IRCoT outperforms OneR for both models and all datasets. IRCoT with 3B model even outperforms OneR with 58X larger GPT3 model showing the value of improved retrieval.

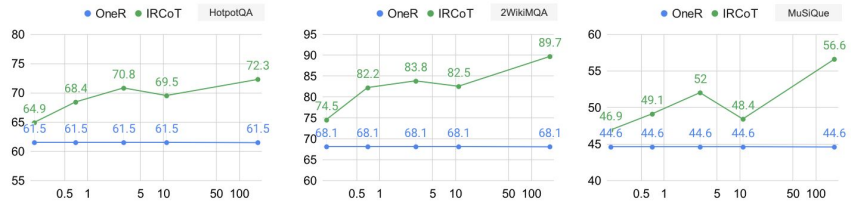


Figure 5: Retrieval recall for OneR (bottom line) and IRCoT (top line) for language models of increasing sizes: Flan-T5-base (0.3B), Flan-T5-large (0.7B), Flan-T5-XL (3B), Flan-T5-XXL (11B), GPT3 (175B) on HotpotQA (left), 2WikiMultihopQA (middle), MuSiQue (right). IRCoT outperforms OneR for all model sizes, including the 0.3B model, and the difference roughly grows with model size. Note that OneR does not interleave any LM in its retrieval and hence has a fixed score.

Other Retrieval Augmented LMs

Shall We Pretrain Autoregressive Language Models with Retrieval?
A Comprehensive Study

Boxin Wang[*†1] Wei Ping[*†2] Peng Xu[*2] Lawrence McAfee[2]

Zihan Liu[2] Mohammad Shoeybi[2] Yi Dong[2] Oleksii Kuchaiev[2]

Bo Li[1] Chaowei Xiao[2,3] Anima Anandkumar[2] Bryan Catanzaro[2]

Model Name	#/ Retrieval Tokens	When to Involve Retrieval	Architecture	Initialization	Re-indexing
RETRO (Borgeaud et al.)	$O(10^{12})$	Pretraining	decoder-only	From Scratch / Pretrained GPT	No
Atlas (Izacard et al.)	$O(10^{9})$	Pretraining	encoder-decoder	Pretrained T5	Yes
REALM (Guu et al.)	$O(10^{9})$	Pretraining	encoder-only	Pretrained BERT	Yes
RAG (Lewis et al.)	$O(10^{9})$	Fine-tuning	encoder-decoder	Pretrained BART	No
DPR (Karpukhin et al.)	$O(10^{9})$	Fine-tuning	encoder-only	Pretrained BERT	No
FiD (Izacard and Grave)	$O(10^{9})$	Fine-tuning	encoder-decoder	Pretrained T5	No
KNN-LM (Khandelwal et al.)	$O(10^{9})$	Inference	decoder-only	Pretrained GPT	No

Table 1: Comparison of different retrieval-augmented models in terms of #/ retrieval tokens, which stage to incorporate retrieval into LMs, the architecture of the backbone LM, whether it requires initialization from the existing LM checkpoint, and whether it requires expensive re-indexing. RETRO is the most scalable retrieval-augmented LM due to its chunk-level retrieval and scalable decoder-only autoregressive LM backbone (Thoppilan et al., 2022; Brown et al., 2020; Smith et al., 2022; Chowdhery et al., 2022) without expensive retrieval index refresh.

Source

Takeaways

- CoT (et al) can be viewed as general context expansion technique used in conjunction with other methods discussed

- Code conditioning provides useful signal

- Use tools like internet search / wikipedia search and combine with CoT techniques
    - Inference v/s training time

Ethical Considerations

On Second Thought, Let's Not Think Step by Step!
Bias and Toxicity in Zero-Shot Reasoning

Omar Shaikh, **Hongxin Zhang**, **William Held**, **Michael Bernstein**, **Diyi Yang**
Stanford University, Shanghai Jiao Tong University, Georgia Institute of Technology
oshaikh@stanford.edu, icefox@sjtu.edu.cn, wheld3@gatech.edu
{mbernst, diyiy}@cs.stanford.edu

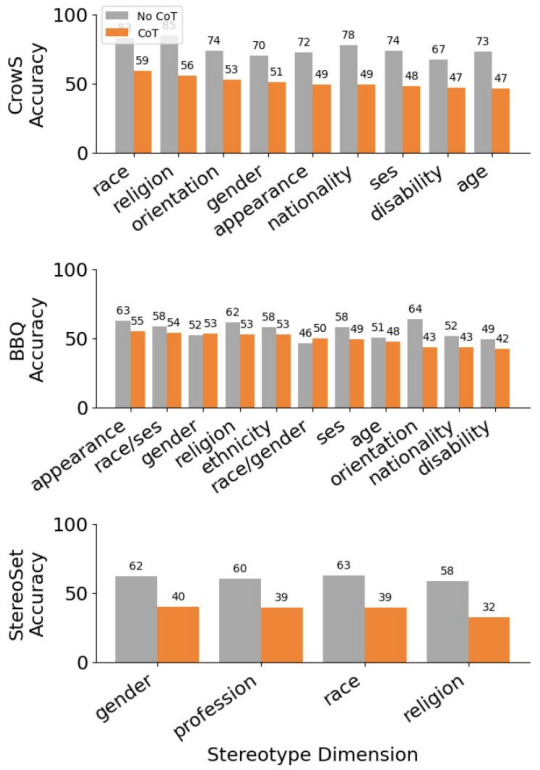


Figure 2: **Accuracy Degredations Across Dimension** for benchmark categories when using `text-davinci-002`. Percentages closer to 100 are better. Categories are sorted by CoT accuracy.

Source

:)

The Cambrian explosion of LM++'s (Language models plus search, retrieval, chain of thought, etc.) will take years to *fully* settle but it'll be much clearer what works and which of the zillion products matter by the end of the year.

Until then, no one knows what's going on.

Source

Thank you!

For any questions, please reach out at gargib2@illinois.edu