

Towards Revealing the Mystery behind Chain of Thought

Gu hao Feng*, Bohang Zhang*, Yuntian Gu*, Haotian Ye*, Di He, Liwei Wang

Peking University

June 16, 2023

Catalogue

- 1 Introduction
- 2 Preliminary
- 3 Results and Intuitions
- 4 Experiments

Catalogue

1 Introduction

2 Preliminary

3 Results and Intuitions

4 Experiments

Capabilities of LLMs

Large Language Models (LLMs) have demonstrated emergent capabilities in various aspects [Bubeck et al., 2023]:

- Generation: translation, summary, composition, ...
- Question answering
- Mathematics
- Coding
- Reasoning, Planning, Decision-making, ...

Autoregressive Transformers

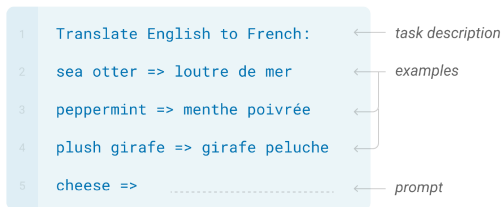
- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].

Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.

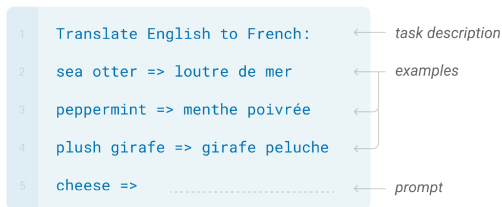
Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.
- The task description along with the task input can be together encoded as a sequence of tokens, called the *prompt*.



Autoregressive Transformers

- Most LLMs follow the autoregressive design paradigm [Radford et al., 2019, Brown et al., 2020, OpenAI, 2023, Zhang et al., 2022, Touvron et al., 2023, Chowdhery et al., 2022, Rae et al., 2021, Scao et al., 2022].
- Main idea: various tasks can be uniformly treated as sequence generation problems.
- The task description along with the task input can be together encoded as a sequence of tokens, called the *prompt*.
- The answer is generated by predicting subsequent tokens conditioned on the prompt in an autoregressive way.



Chain of Thought Prompting (CoT)

- Crucial for tasks involving math or reasoning [Wei et al., 2022, Kojima et al., 2022, Suzgun et al., 2022, Nye et al., 2022, Wies et al., 2023].
- Two typical triggering methods:

(a) 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 X

(b) 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. X

(c) Zero-shot-CoT

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. ✓

(d) 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. ✓

Questions Regarding CoT

- Why is CoT generation so powerful?
- How can these prompts trigger the CoT generation? Can we design better prompting strategies to further exploit the power of LLMs?
- How can CoT emerge in LLMs trained over massive data?

Questions Regarding CoT

- How can we theoretically understand the power of CoT generation?
- How can these prompts trigger the CoT generation? Can we design better prompting strategies to further exploit the power of LLMs?
- How can CoT emerge in LLMs trained over massive data?

We focus on the first aspect by answering two central questions:

- Are there indeed *inherent* limitations of LLMs in directly solving math/reasoning tasks (without CoT)?
- What is the essential reason that CoT generation can bypass the limitations and boost the performance of LLMs?

Catalogue

1 Introduction

2 Preliminary

3 Results and Intuitions

4 Experiments

Autoregressive Transformers

- Input: a sequence of tokens s of length n .
- Initial embedding: $\mathbf{X}^{(0)} = [\mathbf{v}_1 + \mathbf{p}_1, \dots, \mathbf{v}_n + \mathbf{p}_n]^\top \in \mathbb{R}^{n \times d}$, where
 - ▶ each input token s_i is converted to a d -dimensional vector $\mathbf{v}_i = \text{Embed}(s_i) \in \mathbb{R}^d$;
 - ▶ $\mathbf{p}_i \in \mathbb{R}^d$ is the positional embedding.
- Propagation: L Transformer blocks follow, each of which transforms the input by

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} + \text{Attn}^{(l)}(\mathbf{X}^{(l-1)}) + \text{FFN}^{(l)}\left(\mathbf{X}^{(l-1)} + \text{Attn}^{(l)}(\mathbf{X}^{(l-1)})\right),$$

- ▶ $\text{Attn}^{(l)}$ is a multi-head self-attention layer;
- ▶ $\text{FFN}^{(l)}$ is a 2-layer feed forward network with GeLU activation.

$$\text{FFN}^{(l)}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_1^{(l)})\mathbf{W}_2^{(l)}.$$

Autoregressive Transformers

- Multi-head self attention:

$$\text{Attn}^{(l)}(\mathbf{X}) = \sum_{h=1}^H \text{softmax} \left(\mathbf{X} \mathbf{W}_Q^{(l,h)} (\mathbf{X} \mathbf{W}_K^{(l,h)})^\top + \mathbf{M} \right) \mathbf{X} \mathbf{W}_V^{(l,h)} \mathbf{W}_O^{(l,h)},$$

- ▶ The matrix $\mathbf{M} \in \{-\infty, 0\}^{n \times n}$ is a causal mask defined as $M_{ij} = -\infty$ iff $i < j$. This ensures that each position i can only attend to preceding positions $j \leq i$.
- Output: $\mathbf{X}_{n,:}^{(L)} \in \mathbb{R}^d$ is used to predict the token s_{n+1} (via a softmax classifier).
- Autoregressive generation: By concatenating s_{n+1} to the end of the input sequence s , the above process can be repeated. The process continues iteratively until a designated End-of-Sentence token is generated.

Tasks Specification

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023]. Can we gain insights into their power in fundamental math tasks?

Tasks Specification

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023]. Can we gain insights into their power in fundamental math tasks?
- We consider two tasks: arithmetic and equation, which serve as elementary building blocks in solving complex real-world math problems.

Tasks Specification

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023]. Can we gain insights into their power in fundamental math tasks?
- We consider two tasks: arithmetic and equation, which serve as elementary building blocks in solving complex real-world math problems.
- In addition, we know that LLMs exhibit the ability to make decisions. We also consider a powerful decision-making framework: Dynamic Programming (DP). Can LLMs emulate DP algorithms to break down a complex problem into subproblems and tackle them sequentially?

Tasks Specification

- Transformer-based LLMs exhibit surprising math abilities in various aspects [OpenAI, 2023, Bubeck et al., 2023]. Can we gain insights into their power in fundamental math tasks?
- We consider two tasks: arithmetic and equation, which serve as elementary building blocks in solving complex real-world math problems.
- In addition, we know that LLMs exhibit the ability to make decisions. We also consider a powerful decision-making framework: Dynamic Programming (DP). Can LLMs emulate DP algorithms to break down a complex problem into subproblems and tackle them sequentially?
- We use the example of the Longest Increasing Subsequence to illustrate.

Setups and Illustrations

Two setups:

- Directly generate the answer;
- Generate a complete CoT solution.

Let's try both setups on contemporary LLMs first! Two different ways to prompt: “[Direct]” means directly generating the answer, and “[Chain]” means generating intermediate steps.

GD

Below I will ask you a few questions. When a question is prefixed with [Direct], please directly generate the answer WITHOUT giving any intermediate steps. On the contrary, when a question is prefixed with [Chain], please answer the question in details step by step, and give the final answer in the end.

Arithmetic: Direct

GD

[Direct] $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))=?$



The result of the expression $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))$ is -5.

Arithmetic: Chain of Thought



Sure, let's break this equation down following the order of operations, also known as BIDMAS/BODMAS (Brackets, Indices/Orders, Division and Multiplication (from left to right), Addition and Subtraction (from left to right)).

First, we'll resolve operations inside the brackets:

1. Inside the first set of brackets, we have $(7-2)$ which equals 5.
2. Inside the second set of brackets, we have $(5-3)$ which equals 2.
3. Inside the third set of brackets, we have $(1+2)$ which equals 3.

So the equation now looks like this: $3+(7-5\times 2)-(6-4\times 3)$

Next, we'll carry out the multiplication operations:

4. 5×2 equals 10.
5. 4×3 equals 12.

Our equation now becomes: $3+(7-10)-(6-12)$

Next, we'll do the subtraction:

6. $7-10$ equals -3.
7. $6-12$ equals -6.

Now the equation is: $3+(-3)-(-6)$

Finally, we'll do the addition and subtraction:

8. $3+(-3)$ equals 0.
9. $0-(-6)$ equals 6.

So, the result of the expression $3+(7-(7-2)\times(5-3))-(6-4\times(1+2))$ is 6.

Linear Equation: Direct

GD

[Direct] Solve the following linear equation:

$$3x + 3y + 12z = 6;$$

$$2x + 5y + 14z = 7;$$

$$2x + 4y + 15z = 6;$$



The solution to the system of linear equations is $x = 1$, $y = -1$, and $z = 0$.

Linear Equation: Chain of Thought

From the first simplified equation ($y + 2z = 1$), we can express $y = 1 - 2z$. Substitute y into the second simplified equation ($2y + 7z = 2$), we get:

$$2 \cdot (1 - 2z) + 7z = 2$$

$$2 - 4z + 7z = 2$$

$$3z = 0$$

$$z = 0$$

Substitute $z = 0$ back into $y + 2z = 1$, we get:

$$y + 2 \cdot 0 = 1$$

$$y = 1$$

Substitute $y = 1$ and $z = 0$ into the first equation ($x + y + 4z = 2$), we get:

$$x + 1 + 4 \cdot 0 = 2$$

$$x = 2 - 1$$

$$x = 1$$

So, the solution of this system of linear equations is $x = 1$, $y = 1$, and $z = 0$.

Longest Increasing Sub-sequence: Direct

GD

[Direct] What is the length of the longest increasing sub-sequence of the following sequence?

5 10 6 9 4 7 2 3 8



The length of the longest increasing subsequence of the given sequence is 5.

Longest Increasing Sub-sequence: Chain

The comparison and update would look like this:

1. 10 is greater than 5, so we can extend the increasing subsequence ending at 5 to include 10, so the list becomes [1, 2, 1, 1, 1, 1, 1, 1].
2. 6 is greater than 5, so we can extend the increasing subsequence ending at 5 to include 6, the list becomes [1, 2, 2, 1, 1, 1, 1, 1].
3. 9 is greater than 5, 6 and 10, so we can extend the increasing subsequence ending at 6 to include 9, the list becomes [1, 2, 2, 3, 1, 1, 1, 1].
4. 7 is greater than 5, 6, so we can extend the increasing subsequence ending at 6 to include 7, the list becomes [1, 2, 2, 3, 1, 3, 1, 1].
5. 3 is greater than 2, so we can extend the increasing subsequence ending at 2 to include 3, the list becomes [1, 2, 2, 3, 1, 3, 1, 2].
6. 8 is greater than 2, 3, 4, 5, 6, 7, so we can extend the increasing subsequence ending at 7 to include 8, the list becomes [1, 2, 2, 3, 1, 3, 1, 2, 4].

Step 5: The length of the longest increasing subsequence is the maximum value in the list created in step 2. So, after going through the entire sequence, we look for the maximum value in the list which is 4.

So, the length of the longest increasing subsequence of the given sequence is 4.

Catalogue

1 Introduction

2 Preliminary

3 Results and Intuitions

4 Experiments

Problem Formulation

- We consider a simplified setting where all numbers are integers ranging from $\{0, \dots, p-1\}$ and arithmetic operations are performed in the finite field modulo p (p is a prime number).
- $\text{Arithmetic}(n, p)$: the task of evaluating arithmetic expressions (modulo p), where the input length is bounded by n .
- $\text{Equation}(m, p)$: the task of solving linear equations (modulo p) with no more than m variables.
- For DP problems, please refer to our paper :)

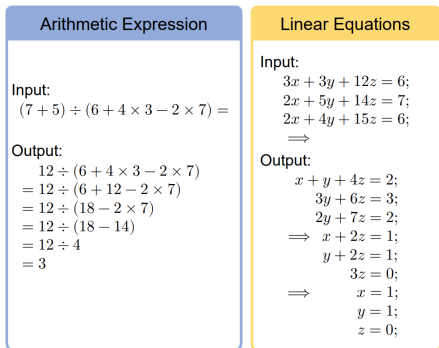


Figure 1: Illustrations of CoT on two math tasks.

Negative Results

Theorem

Assume $\text{TC}^0 \neq \text{NC}^1$. For any prime number p , integer L , and any polynomial Q ,

- there exists a problem size n such that no log-precision autoregressive Transformer with depth L and hidden dimension $d \leq Q(n)$ can solve the problem $\text{Arithmetic}(n, p)$.
 - there exists a problem size m such that no log-precision autoregressive Transformer with depth L and hidden dimension $d \leq Q(m)$ can solve the problem $\text{Equation}(m, p)$.
-
- Our theorems imply that in order to directly output the answers sequentially, the size of the model will grow **super-polynomially** in the input length for both problems.
 - Here TC^0 and NC^1 are two computation complexity classes just like P and NP, and it is widely regarded that $\text{TC}^0 \subsetneq \text{NC}^1$.

How About generating a CoT solution?

Theorem

Fix any prime p . For any integer $n > 0$, there exists an autoregressive Transformer with constant hidden size d (independent of n), depth $L = 5$, and 5 heads in each layer that can generate the CoT solution for all inputs in $\text{Arithmetic}(n, p)$.

Moreover, all parameter values in the Transformer are bounded by $O(\text{poly}(n))$.

Theorem

Fix any prime p . For any integer $m > 0$, there exists an autoregressive Transformer with constant hidden size d (independent of m), depth $L = 5$, and 5 heads in each layer that can generate the CoT solution for all inputs in $\text{Equation}(m, p)$.

Moreover, all parameter values in the Transformer are bounded by $O(\text{poly}(m))$.

Discussions

- How can LLMs equipped with CoT bypass the impossibility results?
 - ▶ This can be understood via the *effective depth* of the Transformer model. Employing CoT creates dependency between output tokens and leads to a significantly deeper model with depth proportional to the length of the CoT solution.
 - ▶ The model can decompose the problem into various subproblems, so that each subproblem can be efficiently and effectively solved with a constant layer model by utilizing the answers (or other relevant information) obtained from previous tokens.
 - ▶ Consequently, CoT generation yields an expressivity far beyond any constant layer models.

Catalogue

1 Introduction

2 Preliminary

3 Results and Intuitions

4 Experiments

Experiments

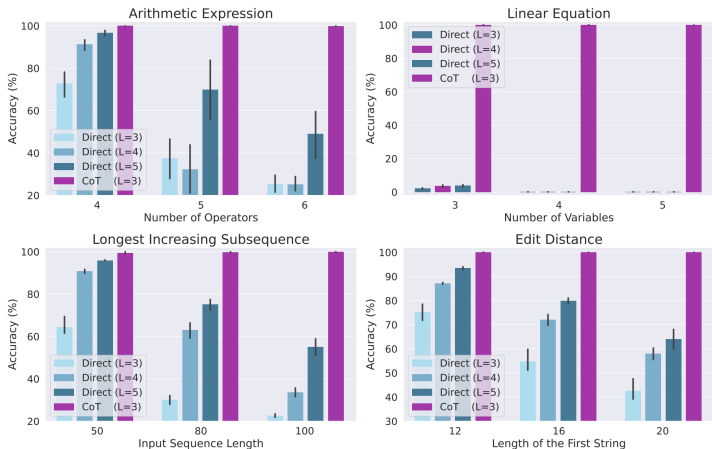


Figure 2: Model performance on different tasks. For all tasks and various difficulty levels, autoregressive Transformers with CoT consistently outperform Transformers trained on direct datasets. In particular, 3-layer Transformers already succeed in these tasks with almost perfect accuracy, while deeper Transformers ($L = 3, 4, 5$) trained on the direct datasets typically fail.

Experiments: Length Extrapolation

Trained on data with number of operators less than 16, and test on longer samples.

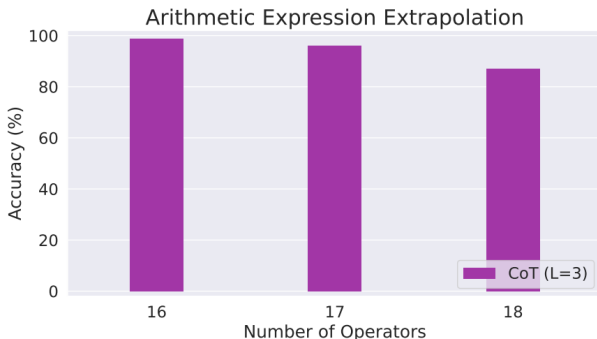


Figure 3: Performance of the length extrapolation experiment, tested on sequences that are longer than those in training.

Extensions

Many theoretical and empirical problems remain to be studied:

- How to make generation “shorter” and “faster” while maintaining the benefit of CoT → architecture improvement.
- How to trigger an appropriate CoT process and delegate sub-problems to plugins / search?
- How can CoT solutions be learnt when there are only limited CoT demonstrations during training? Is it even true?
- Eventually, how to do knowledge verification and avoid hallucination?

References I

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901, 2020.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

References II

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*, 2022.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

References III

- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

References IV

- Noam Wies, Yoav Levine, and Amnon Shashua. Sub-task decomposition enables learning in sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=BrJATVZDWEH>.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Thank You!

<https://haotianye.com>