LLM Reasoning: Key Ideas and Limitations

Denny Zhou
Google DeepMind

April, 2024
My mom is as smart as AI

Mother's Day 2023, my little son wrote:
Self-driving cars

Solve hardest math problems

Superhuman intelligence …

What do you want about AI?
My little expectation about AI

AI should be able to learn from only a few examples, like what humans usually do.
Does machine learning meet this expectation?

Semi-supervised learning
Manifold learning
Sparsity and low rank
Active learning
Bayesian nonparametric
Kernel machines
...
What is missing in machine learning?

Reasoning

Humans can learn from only a few examples because humans can reason.
Let’s start from a toy problem

“Make things as simple as possible but no simpler”

— Albert Einstein
Last Letter Concatenation

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Elon Musk”</td>
<td>“nk”</td>
</tr>
<tr>
<td>“Bill Gates”</td>
<td>“ls”</td>
</tr>
<tr>
<td>“Barack Obama”</td>
<td>?</td>
</tr>
</tbody>
</table>

**Rule:** Take the last letter of each word, and then concatenate them
Solve it by machine learning? Tons of labels needed!

Would you like to call an ML method which needs tons of labels to learn a “trivial” task as AI?
How can this problem be solved by LLMs?
What are **Large Language Models (LLMs)**?

LLM is a “transformer” model trained to predict the next word

![Diagram of LLM processing natural language text](image)

Trained with many sentences, e.g. all texts from the Internet
You can think of training LLMs as training parrots to mimic human languages
Few-shot prompting for last-letter-concatenation

Input

Q: “Elon Musk”
A: “nk”

Q: “Bill Gates”
A: “ls”

Q: “Barack Obama”
A:
Let’s add “reasoning process” before “answer”

Q: “Elon Musk”
A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk" so the output is "nk".

Q: “Bill Gates”
A: the last letter of "Bill" is "l". the last letter of "Gates" is "s". Concatenating "l", "s" leads to "ls". so the output is "ls".

Q: “Barack Obama”
A: the last letter of "Barack" is "k". the last letter of "Obama" is "a". Concatenating "k", "a" leads to "ka", so the output is "ka".

reasoning process
One demonstration is enough, like humans

Q: “Elon Musk”
A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk". so the output is "nk".

Q: “Barack Obama"
A: the last letter of "Barack" is "k", the last letter of "Obama" is "a". Concatenating "k", "a" leads to "ka", so the output is "ka".
Key Idea: Derive the Final Answer through Intermediate Steps

Ling et al 2017 in DeepMind pioneered using natural language rationale to solve math problems by “... derive the final answer through a series of small steps”. Trained a sequence-to-sequence model from scratch.

Problem 1:
Question: Two trains running in opposite directions cross a man standing on the platform in 27 seconds and 17 seconds respectively and they cross each other in 23 seconds. The ratio of their speeds is:
Options: A) 3/7  B) 3/2  C) 3/88  D) 3/8  E) 2/2

Rationale: Let the speeds of the two trains be \(x\) m/sec and \(y\) m/sec respectively. Then, length of the first train = 27\(x\) meters, and length of the second train = 17\(y\) meters. \((27x + 17y) / (x + y) = 23\) → \(27x + 17y = 23x + 23y\) → \(4x = 6y\) → \(x/y = 3/2\)

Correct Option: B

GSM8K: <Problem, Solution, Answer>

Following the work by Ling et al 2017, Cobbe et al 2021 in OpenAI built a much larger math word problem dataset (GSM8K) with natural language rationales, and used it to finetune GPT3

**Problem:** Ali is a dean of a private school where he teaches one class. John is also a dean of a public school. John has two classes in his school. Each class has 1/8 the capacity of Ali’s class which has the capacity of 120 students. What is the combined capacity of both schools?

**Solution:** Ali’s class has a capacity of 120 students. Each of John’s classes has a capacity of 120/8 = 15 students. The total capacity of John’s two classes is 15 students * 2 classes = 30 students. The combined capacity of the two schools is 120 students + 30 students = 150 students.

**Final answer:** 150
Scratchpad for Intermediate Computation

Nye et al 2021 proposed **Scratchpad**: predicting the final output of a program by predicting its intermediate execution result from line to line.

- **Input:**
  2 9 + 5 7

- **Target:**
  
  ```<scratch>
  2 9 + 5 7,  C: 0
  2 + 5, 6 C: 1 # added 9 + 7 = 6 carry 1
  , 8 6 C: 0 # added 2 + 5 + 1 = 8 carry 0
  0 8 6
  </scratch>
  8 6```

---

Chain of Thought Prompting

Self-Consistency

Greatly improves step-by-step reasoning

Motivation

Answer in the greedy output ≠ Maximum likelihood answer
Key Idea: Marginalize **Latent** Reasoning Paths

\[
\begin{align*}
\arg\max \mathbb{P}(\text{answer}|\text{problem}) & \quad \text{(find the answer with the maximum probability)} \\
= \sum_{\text{rationale}} \mathbb{P}(\text{answer, rationale}|\text{problem}) & \quad \text{(sum over all latent reasoning paths)} \\
\approx \frac{\text{frequency of the answer}}{\text{total number of sampled responses}} & \quad \text{(approximate the sum by sampling)} \\
\propto \text{frequency of the answer} & \quad \text{(ignore the common denominator)}
\end{align*}
\]

Thus, \( \arg\max \mathbb{P}(\text{answer}|\text{problem}) \approx \) selecting the most frequent answer.
[Question] Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day?

Random sampled responses from LLM

Response 1: She has 16 - 3 - 4 = 9 eggs left. So she makes $2 * 9 = $18 per day.

Response 2: This means she sells the remainder for $2 * (16 - 4 - 3) = $26 per day.

Response 3: She eats 3 for breakfast, so she has 16 - 3 = 13 left. Then she bakes muffins, so she has 13 - 4 = 9 eggs left. So she has 9 eggs * $2 = $18.

Most frequent answer is: 18
(Not most frequent reasoning path!)
Crushed GSM8K SOTA with only 8 examples
More consistent, more likely to be correct!
How about free-from answers?

Universal Self-Consistency (USC)

Ask LLMs to self-select the most consistent answer!

[Question] Where do people drink less coffee than they do in Mexico?

Response 1: ... Some examples include Japan, China and the United Kingdom.

It is important to note that coffee consumption can vary among individuals within these countries, and preferences can change depending on different factors such as...

Response 2: People in countries like Japan, China, and India typically drink less coffee than they do in Mexico...

Response 3: There are several countries where people generally drink less coffee compared to Mexico. Some of these countries include:

1. Japan:
2. China:
3. Saudi Arabia:
4. India:

The most consistent response: 2
Least-to-Most Prompting

Enable easy-to-hard generalization by decomposition

CoT prompting fails when the test lists are longer (length = 4 and above) than the few-shot examples (length = 2 or 3)

Revisit Last-Letter-Concatenation
Key Idea: Incorporating Reasoning Strategies

Decomposing + Sequential Solving

1. Decompose a complex problem into a list of easier subproblems
2. Solve these subproblems one by one (from least to most complex)
Decomposing and recombining are important operations of the mind.

differently. You decompose the whole into its parts, and you recombine the parts into a more or less different whole.

1. If you go into detail you may lose yourself in details. Too many or too minute particulars are a burden on the mind. They may prevent you from giving sufficient attention to the main point, or even from seeing the main point at all. Think of the man who cannot see the forest for the trees.
Stage 1: Decompose Question into Subquestions

Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?

A: To solve “How many times can she slide before it closes?”, we need to first solve: “How long does each trip take?”

Stage 2: Sequentially Solve Subquestions

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Subquestion 1

Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.

Append model answer to Subquestion 1

Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.

Subquestion 2

Q: How many times can she slide before it closes?

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide 15 ÷ 5 = 3 times before it closes.
Last-Letter-Concatenation (Length Generalization)

<table>
<thead>
<tr>
<th></th>
<th>$L = 4$</th>
<th>$L = 6$</th>
<th>$L = 8$</th>
<th>$L = 10$</th>
<th>$L = 12$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard prompting</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Chain-of-Thought</td>
<td>84.2</td>
<td>69.2</td>
<td>50.2</td>
<td>39.8</td>
<td>31.8</td>
</tr>
<tr>
<td>Least-to-Most</td>
<td>94.0</td>
<td>88.4</td>
<td>83.0</td>
<td>76.4</td>
<td>74.0</td>
</tr>
</tbody>
</table>

Q: “think, machine”
A: The last letter of “think” is “k”. The last letter of “machine” is “e”. Concatenating “k”, “e” leads to “ke”. So, “think, machine” outputs “ke”.

Q: “think, machine, learning”
A: “think, machine” outputs “ke”. The last letter of “learning” is “g”. Concatenating “ke”, “g” leads to “keg”. So, “think, machine, learning” outputs “keg”.
SCAN (Compositional Generalization)

<table>
<thead>
<tr>
<th>Method</th>
<th>Standard prompting</th>
<th>Chain-of-Thought</th>
<th>Least-to-Most</th>
</tr>
</thead>
<tbody>
<tr>
<td>code-davinci-002</td>
<td>16.7</td>
<td>16.2</td>
<td>99.7</td>
</tr>
<tr>
<td>text-davinci-002</td>
<td>6.0</td>
<td>0.0</td>
<td>76.0</td>
</tr>
<tr>
<td>code-davinci-001</td>
<td>0.4</td>
<td>0.0</td>
<td>60.7</td>
</tr>
</tbody>
</table>

SCAN is a task to translate natural language commands to action sequences.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>“look thrice after jump”</td>
<td>JUMP LOOK LOOK LOOK LOOK</td>
</tr>
<tr>
<td>“run left and walk”</td>
<td>TURN_LEFT RUN WALK</td>
</tr>
<tr>
<td>“look opposite right”</td>
<td>TURN_RIGHT TURN_RIGHT LOOK</td>
</tr>
</tbody>
</table>
**CFQ** (Compositional Generalization): Text-to-Code

<table>
<thead>
<tr>
<th></th>
<th>MCD1</th>
<th>MCD2</th>
<th>MCD3</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully Supervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5-base (Herzig et al., 2021)</td>
<td>58.5</td>
<td>27.0</td>
<td>18.4</td>
<td>34.6</td>
</tr>
<tr>
<td>T5-large (Herzig et al., 2021)</td>
<td>65.1</td>
<td>32.3</td>
<td>25.4</td>
<td>40.9</td>
</tr>
<tr>
<td>T5-3B (Herzig et al., 2021)</td>
<td>65.0</td>
<td>41.0</td>
<td>42.6</td>
<td>49.5</td>
</tr>
<tr>
<td>HPD (Guo et al., 2020)</td>
<td>79.6</td>
<td>59.6</td>
<td>67.8</td>
<td>69.0</td>
</tr>
<tr>
<td>T5-base + IR (Herzig et al., 2021)</td>
<td>85.8</td>
<td>64.0</td>
<td>53.6</td>
<td>67.8</td>
</tr>
<tr>
<td>T5-large + IR (Herzig et al., 2021)</td>
<td>88.6</td>
<td>79.2</td>
<td>72.7</td>
<td>80.2</td>
</tr>
<tr>
<td>T5-3B + IR (Herzig et al., 2021)</td>
<td>88.4</td>
<td>85.3</td>
<td>77.9</td>
<td>83.9</td>
</tr>
<tr>
<td>LeAR (Liu et al., 2021)</td>
<td>91.7</td>
<td>89.2</td>
<td>91.7</td>
<td>90.9</td>
</tr>
<tr>
<td><strong>Prompting</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Ours) Dynamic Least-to-Most</td>
<td>94.3</td>
<td>95.3</td>
<td>95.5</td>
<td>95.0</td>
</tr>
</tbody>
</table>

Using only 1% data!

Any other reasoning strategies?
13. Looking back. Even fairly good students, when they have obtained the solution of the problem and written down neatly the argument, shut their books and look for something else. Doing so, they miss an important and instructive phase of the work. By looking back at the completed solution, by reconsidering and reexamining the result and the path that led to it, they could consoli-

The student has now carried through his plan. He has written down the solution, checking each step. Thus, he should have good reasons to believe that his solution is correct. Nevertheless, errors are always possible, especially if the argument is long and involved. Hence, verifications are desirable. Especially, if there is some rapid and intuitive procedure to test either the result or the argument, it should not be overlooked. Can you check the result? Can you check the argument?
Teaching Large Language Models to Self-Debug

Figure 1: **SELF-DDEBUGGING** for iterative debugging using a large language model. At each debugging step, the model first generates new code, then the code is executed and the model explains the code. The code explanation along with the execution results constitute the feedback message, which is then sent back to the model to perform more debugging steps. When unit tests are not available, the feedback can be purely based on code explanation.

Is it possible to make one common prompt for all tasks?

Yes!
Key Idea

Making a big prompt by combining prompts from different tasks, and then applying it to new tasks

Implementation

Too big to load? “Store” them in “weights”! (Instruction tuning)

Instruction tuning: using a huge many-shot prompt stored in the model weights to zero-shot new problems
This explains why LLM chat can answer any user query without requiring inputting demonstration examples
Zero-shot is much more convenient than few-shot, but performs worse

- Any zero-shot method outperforming “Let’s think step by step”?
- Any zero-shot method even outperforming few-shot?
“If you can have a superpower what would it be?”

“I want to have a superpower that can let me have whatever power I want any time!”
When instruction-tuned LLMs can “follow instructions”, why not just “instruct” them to self-generate related examples?
LLMs as Analogical Reasoners

Do you know a related problem? We can scarcely imagine a problem absolutely new, unlike and unrelated to any formerly solved problem; but, if such a problem could exist, it would be insoluble. In fact, when solving a problem, we always profit from previously solved problems, using their result, or their method, or the experience we acquired solving them. And, of course, the problems from which we profit must be in some way related to our present problem. Hence the question: Do you know a related problem?

There is usually no difficulty at all in recalling formerly solved problems which are more or less related
“A mathematician is a person who can find analogies between theorems; a better mathematician is one who can see analogies between proofs and the best mathematician can notice analogies between theories. One can imagine that the ultimate mathematician is one who can see analogies between analogies.”

— Stefan Banach
What is the area of the square with the four vertices at (-2, 2), (2, -2), (-2, -6), and (-6, -2)? Recall a related problem, and then solve this one.

The related problem is finding the distance between two points on a coordinate plane. To find the distance between two points, we use the distance formula: \( d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2} \).

In this problem, we have four points: (-2, 2), (2, -2), (-2, -6), and (-6, -2). Let's choose (-2, 2) and (2, -2) to find the distance \( d \) between them: \( d = \sqrt{(2-(-2))^2 + (-2-2)^2} = \sqrt{4^2 + (-4)^2} = \sqrt{16 + 16} = \sqrt{32} \). We know that the distance between two opposite sides of a square is equal. So, the length of each side of the square is \( \sqrt{32} \). To find the area of the square, we square the length of one side: \( \text{Area} = (\sqrt{32})^2 = 32 \).
Our zero-shot outperforms manual few-shot

<table>
<thead>
<tr>
<th>Prompting Method</th>
<th>GSM8K Accuracy</th>
<th>MATH Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPT3.5-turbo</td>
<td>text-davinci-003</td>
</tr>
<tr>
<td>0-shot</td>
<td>75.0%</td>
<td>14.8%</td>
</tr>
<tr>
<td>0-shot CoT</td>
<td>75.8%</td>
<td>50.3%</td>
</tr>
<tr>
<td>5-shot CoT</td>
<td>76.7%</td>
<td>54.0%†</td>
</tr>
<tr>
<td>Ours: Self-generated Exemplars</td>
<td>77.8%</td>
<td>61.0%†</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prompting Method</th>
<th>Word sorting</th>
<th>Logical deduction five objects</th>
<th>Temporal sequences</th>
<th>Reasoning about colored objects</th>
<th>Formal fallacies</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-shot</td>
<td>66.8%</td>
<td>30.0%</td>
<td>40.4%</td>
<td>50.4%</td>
<td>53.6%</td>
</tr>
<tr>
<td>0-shot CoT</td>
<td>67.6%</td>
<td>35.2%</td>
<td>44.8%</td>
<td>61.6%</td>
<td>55.6%</td>
</tr>
<tr>
<td>3-shot CoT</td>
<td>68.4%</td>
<td>36.4%</td>
<td>58.0%</td>
<td>62.0%</td>
<td>55.6%</td>
</tr>
<tr>
<td>Ours: Self-generated Exemplars</td>
<td>75.2%</td>
<td>41.6%</td>
<td>57.6%</td>
<td>68.0%</td>
<td>58.8%</td>
</tr>
</tbody>
</table>

## Our zero-shot outperforms manual few-shot

<table>
<thead>
<tr>
<th>Promoting Method</th>
<th>GPT3.5-turbo-16k</th>
<th>GPT4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc@1</td>
<td>Acc@10</td>
</tr>
<tr>
<td>0-shot</td>
<td>8%</td>
<td>24%</td>
</tr>
<tr>
<td>0-shot CoT</td>
<td>9%</td>
<td>27%</td>
</tr>
<tr>
<td>3-shot CoT</td>
<td>11%</td>
<td>27%</td>
</tr>
<tr>
<td><strong>Ours: Self-generated Exemplars</strong></td>
<td>13%</td>
<td>25%</td>
</tr>
<tr>
<td><strong>Ours: Self-generated Knowledge + Exemplars</strong></td>
<td>15%</td>
<td>29%</td>
</tr>
</tbody>
</table>

Why outperforms “Let’s think step by step”?
Analogical reasoning automatically “few-shot” any problem!

Why outperforms hand-crafted few-shot prompting?
Generates different related exemples/knowledge for different problems!
Do we really have to prompt LLMs to generate step-by-step (chain-of-thought) reasoning?

No
Chain-of-Thought Reasoning without Prompting

Key observations: (1) Pre-trained LLMs have had responses with step-by-step reasoning among the generations started with the top-k tokens; (2) Higher confidence in decoding the final answer when a step-by-step reasoning path is present.
Question in standard QA format

Q: I have 3 apples, my dad has 2 more apples than me, how many apples do we have in total?
A:

Decoding step 0

top-1: 5
top-2: I
top-3: We
top-4: You
top-5: The

Continue greedy decoding

5 apples

I have 3 apples, my dad has 2 more apples than me, so he has 5 apples. $3 + 5 = 8$. We have 8 apples in total.

We have 5 apples in total.

You have 3 apples, your dad has 2 more apples than you, so he has 5 apples. $3 + 5 = 8$. You have 8 apples in total.

The answer is 5.

uncertain  certain
# Chain-of-Thought Decoding

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>X-Small</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>PaLM-2 Inst-tuned</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM8K</td>
<td>greedy</td>
<td>9.0</td>
<td>14.3</td>
<td>21.0</td>
<td>34.8</td>
<td>67.8</td>
</tr>
<tr>
<td></td>
<td>CoT-decoding</td>
<td><strong>17.7</strong> (+8.7)</td>
<td><strong>35.1</strong> (+20.8)</td>
<td><strong>39.7</strong> (+18.7)</td>
<td><strong>61.5</strong> (+26.7)</td>
<td><strong>81.3</strong> (+13.5)</td>
</tr>
<tr>
<td>MultiArith</td>
<td>greedy</td>
<td>7.5</td>
<td>15.8</td>
<td>36.8</td>
<td>75.0</td>
<td>93.7</td>
</tr>
<tr>
<td></td>
<td>CoT-decoding</td>
<td><strong>34.8</strong> (+27.3)</td>
<td><strong>43.5</strong> (+27.7)</td>
<td><strong>52.5</strong> (+15.7)</td>
<td><strong>86.7</strong> (+11.7)</td>
<td><strong>98.7</strong> (+5.0)</td>
</tr>
<tr>
<td></td>
<td>PaLM-2 Pre-trained</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Small</td>
<td>Medium</td>
<td>Large</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greedy</td>
<td>61.0</td>
<td>55.0</td>
<td>57.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-decoding</td>
<td>65.0 (+4.0)</td>
<td>89.0 (+34.0)</td>
<td>95.0 (+38.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[Year Parity]** Was Nicolas Cage born in an even or odd year?

**Greedy path:**

$k = 0$: Nicolas Cage was born in an **odd** year. (0.117)

**Alternative top-$k$ paths:**

$k = 1$: Even (0.207)

$k = 2$: Odd (0.198)

$k = 3$: 1964, an **even** year. (0.949)

$k = 4$: He was born in an **even** year. (0.0)

...  

$k = 7$: Cage was born in 1964, an **even** year. (0.978)
Limitations
LLMs Can Be Easily Distracted by Irreverent Context

Humans may be easily distracted by irrelevant context

Psychology studies show that irrelevant information may significantly decrease some children and even adults problem-solving accuracy

… inattentive children’s difficulties in problem solving are partially due to an inability to inhibit irrelevant information …

Does this observation hold for LLMs?

LLMs can be easily distracted by irreverent context

Adding irrelevant contexts to GSM8K leads to 20+ points performance drop

Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. Maria's monthly rent is $10. What is Lucy’s bank balance?

Lucy’s bank balance is $65 + $15 - $4 - $10 = $66. The answer is $66.
Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. Maria's monthly rent is $10. What is Lucy’s bank balance? Ignore irrelevant context.

Lucy has $65 in the bank. After making a $15 deposit and then a $4 withdrawal, her bank balance is $76. Maria's monthly rent is not relevant to this question, so it can be ignored. The answer is $76.
LLMs Cannot Self-Correct Reasoning Yet

“People who are unskilled in these domains suffer a dual burden: Not only do these people reach erroneous conclusions and make unfortunate choices, but their incompetence robs them of the metacognitive ability to realize it.”

### Self-Correction Leads to Worse Results

<table>
<thead>
<tr>
<th></th>
<th># calls</th>
<th>GSM8K</th>
<th>CommonSenseQA</th>
<th>HotpotQA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPT-3.5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Prompting</td>
<td>1</td>
<td>75.9</td>
<td>75.8</td>
<td>26.0</td>
</tr>
<tr>
<td>Self-Correct (round 1)</td>
<td>3</td>
<td>75.1</td>
<td>38.1</td>
<td>25.0</td>
</tr>
<tr>
<td>Self-Correct (round 2)</td>
<td>5</td>
<td>74.7</td>
<td>41.8</td>
<td>25.0</td>
</tr>
<tr>
<td><strong>GPT-4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Prompting</td>
<td>1</td>
<td>95.5</td>
<td>82.0</td>
<td>49.0</td>
</tr>
<tr>
<td>Self-Correct (round 1)</td>
<td>3</td>
<td>91.5</td>
<td>79.5</td>
<td>49.0</td>
</tr>
<tr>
<td>Self-Correct (round 2)</td>
<td>5</td>
<td>89.0</td>
<td>80.0</td>
<td>43.0</td>
</tr>
</tbody>
</table>

Too difficult for LLMs to self realize their mistakes
**Reported Improvements Need Oracle Labels!**

<table>
<thead>
<tr>
<th></th>
<th>GSM8K</th>
<th>CommonSenseQA</th>
<th>HotpotQA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPT-3.5</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StandardPrompting</td>
<td>75.9</td>
<td>75.8</td>
<td>26.0</td>
</tr>
<tr>
<td>Self-Correct (Oracle)</td>
<td>84.3</td>
<td>89.7</td>
<td>29.0</td>
</tr>
<tr>
<td><strong>GPT-4</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StandardPrompting</td>
<td>95.5</td>
<td>82.0</td>
<td>49.0</td>
</tr>
<tr>
<td>Self-Correct (Oracle)</td>
<td>97.5</td>
<td>85.5</td>
<td>59.0</td>
</tr>
</tbody>
</table>

Oracle: Let LLMs self correct only when the answer is wrong
Multi-Agent Debate? Worse than Self-Consistency!

<table>
<thead>
<tr>
<th></th>
<th># responses</th>
<th>GSM8K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Prompting</td>
<td>1</td>
<td>76.7</td>
</tr>
<tr>
<td>Self-Consistency</td>
<td>3</td>
<td>82.5</td>
</tr>
<tr>
<td>Multi-Agent Debate (round 1)</td>
<td>6</td>
<td>83.2</td>
</tr>
<tr>
<td>Self-Consistency</td>
<td>6</td>
<td>85.3</td>
</tr>
<tr>
<td>Multi-Agent Debate (round 2)</td>
<td>9</td>
<td>83.0</td>
</tr>
<tr>
<td>Self-Consistency</td>
<td>9</td>
<td><strong>88.2</strong></td>
</tr>
</tbody>
</table>
Need external oracle feedback to make LLM self-correction work. Self-debug naturally leverages unit tests in code generation tasks.
Premise Order Matters in LLM Reasoning

For simple logic inference tasks, **permuting the premise order** causes 30+ points performance drop across all LLMs:

1. Presenting “If A then B” before “If B then C” generally achieves a higher accuracy compared to the reversed order.

2. The performance gap becomes more significant when the number of premises increases.
Theoretic Analysis

“There is nothing more practical than a good theory.”

— Kurt Lewin
What learning algorithm is in-context learning?

Transformer-based in-context learners implement standard learning algorithms, such as gradient descent, implicitly, by encoding smaller models in their activations, and updating as new examples come.
Chain of thought empowers transformers to solve inherently serial problems

Transformer deriving final answers with intermediate steps can solve any inherently serial problems as long as its depth exceeds a threshold.

Transformer directly generating answers either requires a huge depth to solve or cannot solve them at all.

CoT here means CoT responses, not CoT prompting.

Summary

Key Ideas
- Step-by-step/chain-of-thought
- Self-consistency/marginalization
- Incorporating reasoning strategies
  - Decompose (least-to-most)
  - Self-examine (self-debug)
  - Analogical reasoning

Limitations
- Easily distracted by irrelevant context
- Cannot self-correct reasoning yet
- Premise order matters

Few-shot prompting → Zero-shot prompting → No prompting
What is next?

“If I were given one hour to save the planet, I would spend 59 minutes defining the problem and one minute resolving it.”

— Albert Einstein
Conference on Language Modeling (COLM)
Thank You

https://twitter.com/denny_zhou

https://dennyzhou.github.io/

https://scholar.google.com/citations?user=UwLsYw8AAAAJ&hl=en