Teach Language Models to Reason

Denny Zhou

Google DeepMind

December, 2023
What do you expect from AI?

- Self-driving cars
- Digital assistant
- Solving hardest math problems
  ...
As a machine learning researcher, my dream is to achieve

Accuracy 100%

by learning 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
Transfer learning
Metalearning
Bayesian nonparametric
Kernel machines
...

MISSION FAILED
What is missing in machine learning?

Reasoning

Humans can learn from only a few examples because humans can reason...
We have found a simple way to solve reasoning:

Teach language models to reason, like teaching kids
Let’s start from a toy problem
Toy problem: 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 to solve this problem with LLMs?
What are Large Language Models (LLMs)?

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

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

Q: “Elon Musk”
A: “nk”

Q: “Bill Gates”
A: “Is”

Q: “Barack Obama”
A:
Q: “Elon Musk”
A: “nk”

Q: “Bill Gates”
A: “Is”

Q: “Barack Obama”
A: “ma”

FAILED
Q: “Elon Musk”
A: “nk”

Q: “Bill Gates”
A: “Is”

Q: “Barack Obama”
A: “ma”

How about adding more examples?
Q: "Elon Musk"
A: "nk"

Q: "Bill Gates"
A: "Is"

Q: "Steve Jobs"
A: "es"

Q: "Larry Page"
A: "ye"

Q: "Jeff Bezos"
A: "fs"

Q: "Barack Obama"
A: "ma"
Why we created the last-letter-concatenation task?

- Machine learning fail
- Few-shot prompting fail
- Trivial for humans
Chain-of-Thought (CoT) Prompting

CoT: Adding “thought” 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". thought

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:
CoT: Adding “thought” 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". thought

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".
One demonstration is enough, as 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".
Standard few-shot prompting
<input, output>

Chain-of-thought prompting
<input, thought, output>

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

Wei et al. Chain-of-thought prompting elicits reasoning in large language models. Jan, 2022
Can LLMs solve math word problems?

Elsa has 3 apples. Anna has 2 more apples than Elsa. How many apples do they have together?

They have 5 apples together.
Let’s see how CoT magically works

**Demonstration example with “thought”**

There were 3 pizzas in total at the pizza shop. A customer bought one pizza. How many pizzas are left?
There are 3 pizzas in total and a customer bought one. So there are $3 - 1 = 2$ pizzas left. So the answer is 2.

Elsa has 3 apples. Anna has 2 more apples than Elsa. How many apples do they have together?

Elsa has 3 apples and Anna has 2 more apples than Elsa. So Anna has $3 + 2 = 5$ apples. So Elsa and Anna have $3 + 5 = 8$ apples together. So the answer is 8.
CoT for any task

**StrategyQA**

Q: Yes or no: Would a pear sink in water?

A: The density of a pear is about 0.6 g/cm^3, which is less than water. Thus, a pear would float. So the answer is no.

**Date Understanding**

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

**Sports Understanding**

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

**SayCan (Instructing a robot)**

Human: How would you bring me something that isn’t a fruit?

Explanation: the user wants something to eat that isn’t a fruit. An energy bar is not a fruit, so I will bring the user an energy bar.

Plan: 1. find(energy bar) 2. pick(energy bar) 3. find(user) 4. put(energy bar) 5. done().

**Last Letter Concatenation**

Q: Take the last letters of the words in "Lady Gaga" and concatenate them.

A: The last letter of “Lady” is “y”. The last letter of “Gaga” is “a”. Concatenating them is “ya”. So the answer is ya.

**Coin Flip (state tracking)**

Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.
100x - 1000x data efficient than supervised sota in the literature
Ling et al 2017 has proposed using natural language rationale to solve math word problems: "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. \((27x + 17y) / (x + y) = 23 \Rightarrow 27x + 17y = 23x + 23y \Rightarrow 4x = 6y \Rightarrow x/y = 3/2\)

**Correct Option:** B

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 using the dataset to finetuned 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
Nye et al 2021 propose **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
```
Self-Consistency

Greatly improves chain-of-thought prompting

Figure 1: The self-consistency method contains three steps: (1) prompt a language model using example chains of thought; (2) sample from the language model’s decoder to generate a diverse set of reasoning paths; and (3) choose the most consistent answer using the majority/plurality vote.
Crushed GSM8K SOTA with only 8 examples
Crushed GSM8K SOTA with only 8 examples

<table>
<thead>
<tr>
<th></th>
<th>GSM8K Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fintuned GPT-3</td>
<td>33</td>
</tr>
<tr>
<td>Fintuned GPT-3 + verifier</td>
<td>55</td>
</tr>
<tr>
<td>PaLM + CoT</td>
<td>58</td>
</tr>
<tr>
<td>PaLM + CoT + SC</td>
<td>75</td>
</tr>
<tr>
<td>PaLM-2 + CoT + SC</td>
<td>92</td>
</tr>
</tbody>
</table>
How many more examples are needed for finetuning to be comparable to CoT + SC?

“We perform finetuning… GSM8K … it appears likely that the 175B model would require at least two additional orders of magnitude of training data to reach an 80% solve rate.”

Solve high school math problems

- Finetuning PaLM with math data
- SC + CoT solves 50%!
- Non-math graduate students solve: 40%

---

**Question:** A particle moves so that it is at \((3 \sin \frac{t}{4}, 3 \cos \frac{t}{4})\) at time \(t\). Find the speed of the particle, measured in unit of distance per unit of time.

**Target:** The parametric curve \((x, y) = (3 \sin \frac{t}{4}, 3 \cos \frac{t}{4})\) describes a circle with radius 3. Furthermore, it makes a full revolution at time \(t = 8\pi\). (figure) Therefore, the speed of the particle is \(\frac{\pi}{8} = \frac{3}{4}\).

**Model output:** The speed is the magnitude of the velocity vector. Therefore, we must first find the velocity vector, which is the derivative of the position vector. We have

\[
\vec{v} = \frac{d}{dt} \left( 3 \sin \frac{t}{4}, 3 \cos \frac{t}{4} \right) = 3 \left( \cos \frac{t}{4}, -\sin \frac{t}{4} \right) \frac{1}{4} = \left( \frac{3}{4} \cos \frac{t}{4}, -\frac{3}{4} \sin \frac{t}{4} \right).
\]

Therefore, the magnitude of the velocity vector is

\[
|\vec{v}| = \sqrt{\left( \frac{3}{4} \cos \frac{t}{4} \right)^2 + \left( -\frac{3}{4} \sin \frac{t}{4} \right)^2} = \sqrt{\frac{9}{16} \cos^2 \frac{t}{4} + \frac{9}{16} \sin^2 \frac{t}{4}}
\]

\[
= \frac{3}{4} \sqrt{\cos^2 \frac{t}{4} + \sin^2 \frac{t}{4}} = \frac{3}{4}.
\]

Therefore, the speed of the particle is \(\frac{3}{4}\).

---

<table>
<thead>
<tr>
<th>Method</th>
<th>AddSub</th>
<th>MultiArith</th>
<th>ASDiv</th>
<th>AQuA</th>
<th>SVAMP</th>
<th>GSM8K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous SoTA</td>
<td>94.9a</td>
<td>60.5a</td>
<td>75.3b</td>
<td>37.9c</td>
<td>57.4d</td>
<td>35e / 55g</td>
</tr>
<tr>
<td>UL2-20B</td>
<td>18.2</td>
<td>10.7</td>
<td>16.9</td>
<td>23.6</td>
<td>12.6</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>24.8 (+6.6)</td>
<td>15.0 (+4.3)</td>
<td>21.5 (+4.6)</td>
<td>26.9 (+3.3)</td>
<td>19.4 (+6.8)</td>
<td>7.3 (+3.2)</td>
</tr>
<tr>
<td>LaMDA-137B</td>
<td>52.9</td>
<td>51.8</td>
<td>49.0</td>
<td>17.7</td>
<td>38.9</td>
<td>17.1</td>
</tr>
<tr>
<td></td>
<td>63.5 (+10.6)</td>
<td>75.7 (+23.9)</td>
<td>58.2 (+9.2)</td>
<td>26.8 (+9.1)</td>
<td>53.3 (+14.4)</td>
<td>27.7 (+10.6)</td>
</tr>
<tr>
<td>PaLM-540B</td>
<td>91.9</td>
<td>94.7</td>
<td>74.0</td>
<td>35.8</td>
<td>79.0</td>
<td>56.5</td>
</tr>
<tr>
<td></td>
<td>93.7 (+1.8)</td>
<td>99.3 (+4.6)</td>
<td>81.9 (+7.9)</td>
<td>48.3 (+12.5)</td>
<td>86.6 (+7.6)</td>
<td>74.4 (+17.9)</td>
</tr>
<tr>
<td>GPT-3</td>
<td>57.2</td>
<td>59.5</td>
<td>52.7</td>
<td>18.9</td>
<td>39.8</td>
<td>14.6</td>
</tr>
<tr>
<td>Code-davinci-001</td>
<td>67.8 (+10.6)</td>
<td>82.7 (+23.2)</td>
<td>61.9 (+9.2)</td>
<td>25.6 (+6.7)</td>
<td>54.5 (+14.7)</td>
<td>23.4 (+8.8)</td>
</tr>
<tr>
<td>GPT-3</td>
<td>89.4</td>
<td>96.2</td>
<td>80.1</td>
<td>39.8</td>
<td>75.8</td>
<td>60.1</td>
</tr>
<tr>
<td>Code-davinci-002</td>
<td>91.6 (+2.2)</td>
<td><strong>100.0 (+3.8)</strong></td>
<td><strong>87.8 (+7.6)</strong></td>
<td><strong>52.0 (+12.2)</strong></td>
<td><strong>86.8 (+11.0)</strong></td>
<td><strong>78.0 (+17.9)</strong></td>
</tr>
</tbody>
</table>

"Self-consistency + chain-of-thought" crushed SOTA by large margin
Why does self-consistency work? Marginalization!

\[
\text{arg max } \mathbb{P}(\text{answer}|\text{problem}) \quad \text{(find the answer with the maximum probability)}
\]

\[
= \sum_{\text{rationale}} \mathbb{P}(\text{answer}, \text{rationale}|\text{problem}) \quad \text{(sum over all latent reasoning paths)}
\]

\[
\approx \frac{\text{frequency of the answer}}{\text{total number of sampled responses}} \quad \text{(approximate the sum by sampling)}
\]

\[
\propto \text{frequency of the answer} \quad \text{(ignore the common denominator)}
\]

Thus, \( \text{arg max } \mathbb{P}(\text{answer}|\text{problem}) \approx \) selecting the most frequent answer.
Self-consistency is the empirical implementation of marginalization. Don’t be superficial to interpret it as majority voting!
When there is no reasoning path, we don't need self-consistency, since we can directly choose the most likely answer using $P(Y|X)$!
More consistent, more likely to be correct!
Universal Self-Consistency (USC) for Any Task

Just ask LLMs to select the most consistent response based on majority consensus!

Universal Self-Consistency (USC) for Any Task

- USC consistently improves the performance on free-form generation tasks, like summarization, where SC is inapplicable.
- For code generation, USC matches the performance of execution-based SC that selects the code with the most consistent execution outputs, while USC does not require code execution.
- For math reasoning, USC matches the performance of SC while eliminating the need of parsing answers to aggregate.
Least-to-Most Prompting

Enable easy-to-hard generalization by decomposition

CoT fails to generalize to harder problems

Last-letter-concatenation task with list longer than demonstration which is limited to 3 words:

<table>
<thead>
<tr>
<th>Method</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>89.4</td>
<td>75.0</td>
<td>51.8</td>
<td>39.8</td>
<td>33.6</td>
</tr>
</tbody>
</table>

CoT’s performance greatly drops as length increases
Key Idea of Least-to-Most Prompting

1. **Decompose** a complex problem into a list of easier subproblems
2. Sequentially **solve these subproblems** (from least to most complex)

Least-to-Most Prompting = Planning/Sketch + Reasoning
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.
Solve math problems by decomposition

Q: Elsa has 3 apples. Anna has 2 more apples than Elsa. How many apples do they have together?

A: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do Elsa and Anna have together?

1. Anna has 2 more apples than Elsa. So Anna has $2 + 3 = 5$ apples.
2. Anna has 5 apples. Elsa and Anna have $3 + 5 = 8$ apples together.

The answer is 8.

Q: Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?

A:
Q: Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?

A: Let's break down this problem: 1. How much does Eliza earn for the first 40 hours she works each week? 2. How much does Eliza earn in overtime pay? 3. How much does Eliza earn in total for the week?

1. Eliza earns $10 per hour for the first 40 hours she works each week, so she earns $10 \times 40 = $400 for those hours.
2. Eliza earns 1.2 times her regular hourly rate for overtime hours, so she earns 1.2 \times $10 = $12 per overtime hour. Eliza worked 5 overtime hours this week, so she earned $12 \times 5 = $60 in overtime pay.
3. Eliza's total earnings for the week are $400 + $60 = $460.

The answer is: $460
Q: Can sunlight travel to the deepest part of the Black Sea?

A: Let's breakdown this problem: "What is the maximum depth of the Black Sea?", "How deep can sunlight penetrate a sea?", "Can sunlight travel to the deepest part of the Black Sea?"

1. The Black Sea has a maximum depth of 2,212 meters.
2. Sunlight does not penetrate water below 1000 meters.
3. 1000 meters is less than 2,212 meters. So sunlight cannot travel to the deepest part of the Black Sea

Then answer is: false.

Q: Did Aristotle use a laptop?
Commonsense reasoning by decomposition (Cont’)

... 

A: Let's breakdown this problem: "When was the laptop invented?", "When did Aristotle live?", "Did Aristotle Use a Laptop?"

1. The laptop was invented in 1981.
2. Aristotle lived from 384 BC to 322 BC.
3. 1981 is greater than 322 BC. So Aristotle did not use a laptop. The answer is: false.
Last-letter task (length generalization)

<table>
<thead>
<tr>
<th>Method</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>89.4</td>
<td>75.0</td>
<td>51.8</td>
<td>39.8</td>
<td>33.6</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>

Table 3: Accuracies of different prompting methods with code-davinci-002 on the last-letter-concatenation task with the length of lists increasing from 4 to 12. All the methods are 2-shot.
Table 1: Least-to-most prompt context (decomposition) for the last-letter-concatenation task. It can decompose arbitrary long lists into sequential sublists with an accuracy of 100%.

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”.

Teach LLMs the trick from n to n+1

Table 2: Least-to-most prompt context (solution) for the last-letter-concatenation task. The two exemplars in this prompt actually demonstrate a base case and a recursive step.
**SCAN** (compositional generalization): text-to-actions

<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>

Table 8: Accuracies (%) of different prompting methods on the test set of SCAN under length split. The results of `text-davinci-002` are based on a random subset of 100 commands.
**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!

Table 1: Test accuracy across the MCD splits for the CFQ dataset.

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 using it for any task

Magic

Any task: including tasks which are no even seen

Implementation

Too big to load? “Store” them in “weights”! (Instruction tuning!)
FLAN2: Finetune PaLM with 1800+ tasks

Parse out all of the names from this message.

Then, sort them in alphabetical order by first name. Also add "Quoc Le" to the list.
This is zero-shot!

Parse out all of the names from this message.

Then, sort them in alphabetical order by first name. Also add "Quoc Le" to the list.

Output

Denny Zhou, Hyung Won Chung, Jason Wei, Quoc Le, Xuezhi Wang, Yi Tay
Pretraining (next-token-prediction) + Instruction tuning → ChatLLM

Chat is nothing but zero-shot prompting!
Zero-Shot
LLMs as Optimizers

Can LLMs solve math word problems?

Elsa has 3 apples. Anna has 2 more apples than Elsa. How many apples do they have together?

They have 5 apples together.
Let’s think step by step: zero-shot CoT

Elsa has 3 apples. Anna has 2 more apples than Elsa. How many apples do they have together? Let’s think step by step.

Step 1: Elsa has 3 apples.

Step 2: Anna has 2 more apples than Elsa.

Step 3: To find out how many apples they have together, we need to add the number of apples that Elsa has (3) to the number of apples that Anna has (2 + 3 = 5).

They have 8 apples together.

Kojima, Takeshi, et al. Large language models are zero-shot reasoners. NeurIPS (2022)
Anything better than “Let’s think step by step”? 
Let LLMs optimize it!

Text: Let’s figure it out! **Score:** 61

Text: Let’s solve the problem. **Score:** 63

(... more texts and scores ...)

Write your new text that is different from the old ones and has a score as high as possible.
Results on GSM8K w/ PaLM

“Let’s think step by step” 71.8%

“Take a deep breath and work on this problem step-by-step” 80.2% (found by LLM optimization)
LLMs as Analogical Reasoners

Instead of manually making few-shot CoT prompts, why not let LLMs automatically generate related examples?
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
**Model Input**

**Q:** What is the area of the square with the four vertices at (-2, 2), (2, -2), (-2, -6), and (-6, -2)?

**Model Output**

**# Relevant exemplars:**
Q: What is the area of the square with a side length of 5?
A: The area of a square is found by squaring the length of its side. So, the area of this square is $5^2 = 25$. ...

**# Solve the initial problem:**
To find the area of the square, we need to find the side length. The length is ... $\sqrt{(2 - (-2))^2 + (-2 - 2)^2} = \sqrt{32}$. So, the area of the square is $(\sqrt{32})^2 = 32$. ...
<table>
<thead>
<tr>
<th>Prompting Method</th>
<th>GSM8K Accuracy</th>
<th></th>
<th>MATH Accuracy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPT3.5-turbo</td>
<td>text-davinci-003</td>
<td>PaLM2</td>
<td>GPT3.5-turbo</td>
</tr>
<tr>
<td>0-shot</td>
<td>75.0%</td>
<td>14.8%</td>
<td>60.8%</td>
<td>33.0%</td>
</tr>
<tr>
<td>0-shot CoT</td>
<td>75.8%</td>
<td>50.3%</td>
<td>78.2%</td>
<td>33.9%</td>
</tr>
<tr>
<td>5-shot CoT</td>
<td>76.7%</td>
<td>54.0%</td>
<td>80.7%</td>
<td>34.9%</td>
</tr>
<tr>
<td>Ours: Self-generated Exemplars</td>
<td><strong>77.8%</strong></td>
<td><strong>61.0%†</strong></td>
<td><strong>81.7%</strong></td>
<td><strong>37.3%</strong></td>
</tr>
</tbody>
</table>
Your goal is to write Python3 code to solve competitive programming problems. Given a problem, explain the core concepts in it and provide other relevant problems. Then solve the original problem.

# Problem: You are given a sequence $a_1, a_2, \ldots, a_n$. Each element of $a$ is 1 or 2. Find out if an integer $k$ exists so that the following conditions are met. $1 \leq k \leq n-1$, and $a_1 \cdot a_2 \ldots a_k = a_{k+1} \ldots a_n$. If there exist multiple $k$ that satisfy the given condition, print the smallest.

# Instruction:

## Algorithms: Identify the core concepts or algorithms used to solve the problem.

## Tutorial: Write a tutorial about these algorithms.

## Example Problems: Provide three examples of relevant competitive programming problems that involve these algorithms. For each problem, describe the problem, explain the solution in detail, and then write the correct Python3 code.

## Python3 code to solve the original problem: - Explanation of the solution: - Python3 code to solve the problem:
<table>
<thead>
<tr>
<th>Prompting 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><strong>15%</strong></td>
<td><strong>29%</strong></td>
</tr>
</tbody>
</table>
Surprising findings

- The automatically generated problems are usually easier than the problem to solve
- The automatically generated problems may contain mistakes

Despite these, the analogical method still works so well!
Why LLMs can reason?
Reasoning as an emergent behavior

- Reasoning emerges from next-token-prediction pretraining
  - CoT is discovered, not manually designed
  - Sharp reasoning performance transition vs (#parameters, #tokens)
- Distill big models to small ones? Check performance on reasoning tasks!
“How to make parrots intelligent?”

“How to scale up!”

https://twitter.com/denny_zhou/status/159145184723932544?s=20
Toward understanding in-context learning

- Transformer models are meta-learners: implicitly learned training algorithms (like gradient descent) from pretraining.
- In the inference time, transformer models implicitly built the prediction model from the inputs and then predict.

Summary

- **Chain-of-thought**: add “thought” before final “answer”
- **Self-consistency**: sample repeatedly, and select the most frequent answer
- **Least-to-most**: decompose to subproblems and solve them one by one
- **Instruction finetuning**: enable zero-shot / chat
- **Auto prompting** (LLMs as optimizers, LLMs as analogical reasoners)
- **Emergence of LLM reasoning and theory**
What is next?
A model with language understanding and reasoning opens a door to infinite possibilities
Conference on Language Modeling (COLM)

https://colmweb.org
Thank You

https://twitter.com/denny_zhou

https://dennyzhou.github.io/

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