Natural Language Processing with Deep Learning

CS224N/Ling284

Shikhar Murty

Lecture 14: Reasoning and Agents
Lecture Plan

Lecture 14: Reasoning and Agents

1. Reasoning in Language Models [35 mins]
2. Mini-break [5 mins]
3. Language Model Agents [40 mins]

• Announcements
  • Project Milestone due on Wed May 22\textsuperscript{nd} at 4:30 pm
  • Your Project Mentors have already reached out to you (If not, let us know via Ed!)
  • Guest lectures on May 21\textsuperscript{st} and May 28\textsuperscript{th}: Students get 0.75\% per guest lecture for attending live or writing a reaction paragraph (More details will be on Ed)
Reasoning
(with Large Language Models)
What is Reasoning?

Using **facts** and **logic** to arrive at an answer
What is Reasoning?

Using **facts** and **logic** to arrive at an answer

**Deductive Reasoning:** Use logic to go from premise to firm conclusion

- Premise: All mammals have kidneys
- Premise: All whales are mammals
- Conclusion: All whales have kidneys
What is Reasoning?

Using *facts* and *logic* to arrive at an answer

**Deductive Reasoning:** Use logic to go from premise to firm conclusion

<table>
<thead>
<tr>
<th>Premise: All mammals have kidneys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premise: All whales are mammals</td>
</tr>
<tr>
<td>Conclusion: All whales have kidneys</td>
</tr>
</tbody>
</table>

**Inductive Reasoning:** From observation, predict a likely conclusion

<table>
<thead>
<tr>
<th>Observation: When we see a creature with wings, it is usually a bird</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observation: We see a creature with wings.</td>
</tr>
<tr>
<td>Conclusion: The creature is likely to be a bird.</td>
</tr>
</tbody>
</table>
What is Reasoning?

Using **facts** and **logic** to arrive at an answer

**Deductive Reasoning:** Use logic to go from premise to firm conclusion

- **Premise:** All mammals have kidneys
- **Premise:** All whales are mammals
- **Conclusion:** All whales have kidneys

**Inductive Reasoning:** From observation, predict a likely conclusion

- **Observation:** When we see a creature with wings, it is usually a bird
- **Observation:** We see a creature with wings.
- **Conclusion:** The creature is likely to be a bird

**Abductive Reasoning:** From observation, predict the most likely explanation

- **Observation:** The car cannot start and there is a puddle of liquid under the engine.
- **Likely Explanation:** The car has a leak in the radiator
Reasoning: Formal vs Informal

**Formal Reasoning:** Follows formal rules of logic along with axiomatic knowledge to derive conclusions.

**Informal Reasoning:** Uses intuition, experience, common sense to arrive at answers.

For most of this lecture, by “reasoning” we mean informal deductive reasoning, often involving multiple steps.
Large Language models are **REALLY GOOD** at predicting plausible continuations of text, that respect constraints in the input, and align well with human preferences.

**Question:** Can current LLMs reason?
Reasoning in Large Language Models: prompting

Chain-of-thought prompting:

![Diagram showing comparison between Standard Prompting and Chain-of-Thought Prompting](image)

Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Source: Wei et al. 2023
Reasoning in Large Language Models: prompting

Zero-shot CoT prompting:

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16/2 = 8 golf balls. Half of the golf balls are blue. So there are 8/2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Let’s think step by step.

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓
CoT with “Self-consistency”: Replace greedy decoding with an ensemble of samples...

Main idea: correct reasoning processes have greater agreement than incorrect processes.

Source: Wang et al. 2023
## Reasoning in Large Language Models: prompting

<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>94.9(^a)</td>
<td>60.5(^a)</td>
<td>75.3(^b)</td>
<td>37.9(^c)</td>
<td>57.4(^d)</td>
<td>35(^e) / 55(^g)</td>
</tr>
<tr>
<td>UL2-20B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-promoting Self-consistency</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-promoting Self-consistency</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-promoting Self-consistency</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>
</tbody>
</table>

Out-performs regular CoT on a variety of benchmarks

Source: Wang et al. 2023
## Reasoning in Large Language Models: prompting

<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>94.9&lt;sup&gt;a&lt;/sup&gt;</td>
<td>60.5&lt;sup&gt;a&lt;/sup&gt;</td>
<td>75.3&lt;sup&gt;b&lt;/sup&gt;</td>
<td>37.9&lt;sup&gt;c&lt;/sup&gt;</td>
<td>57.4&lt;sup&gt;d&lt;/sup&gt;</td>
<td>35&lt;sup&gt;e&lt;/sup&gt; / 55&lt;sup&gt;g&lt;/sup&gt;</td>
</tr>
<tr>
<td>UL2-20B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-prompting</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>Self-consistency</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-prompting</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>Self-consistency</td>
<td>63.5 (+10.6)</td>
<td>75.7 (+23.9)</td>
<td>58.2 (+9.1)</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoT-prompting</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>Self-consistency</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>
</tbody>
</table>

Out-performs regular CoT on a variety of benchmarks

### Source: Wang et al. 2023

<table>
<thead>
<tr>
<th>Method</th>
<th>GSM8K</th>
<th>MultiArith</th>
<th>SVAMP</th>
<th>ARC-e</th>
<th>ARC-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoT (Wei et al., 2022)</td>
<td>17.1</td>
<td>51.8</td>
<td>38.9</td>
<td>75.3</td>
<td>55.1</td>
</tr>
<tr>
<td>Ensemble (3 sets of prompts)</td>
<td>18.6 ± 0.5</td>
<td>57.1 ± 0.7</td>
<td>42.1 ± 0.6</td>
<td>76.6 ± 0.1</td>
<td>57.0 ± 0.2</td>
</tr>
<tr>
<td>Ensemble (40 prompt permutations)</td>
<td>19.2 ± 0.1</td>
<td>60.9 ± 0.2</td>
<td>42.7 ± 0.1</td>
<td>76.9 ± 0.1</td>
<td>57.0 ± 0.1</td>
</tr>
<tr>
<td>Self-Consistency (40 sampled paths)</td>
<td>27.7 ± 0.2</td>
<td>75.7 ± 0.3</td>
<td>53.3 ± 0.2</td>
<td>79.3 ± 0.3</td>
<td>59.8 ± 0.2</td>
</tr>
</tbody>
</table>

Self-consistency is doing more than simple ensembling

Source: Wang et al. 2023
Reasoning in Large Language Models: prompting
Problem decomposition with Least-to-Most prompting

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?”

Source: Zhou et al. 2023
Reasoning in Large Language Models: prompting
Problem decomposition with Least-to-Most prompting

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 slide closes in 15 minutes. How many times can she slide before it closes?

Language Model

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

Subquestion 1

Q: How long does each trip take?

Language Model

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?

Language Model

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide 15 / 5 = 3 times before it closes.

Source: Zhou et al. 2023
Reasoning in Large Language Models: prompting
Least-to-Most prompting for Math reasoning

Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together?
A: Anna has 2 more apples than Elsa, so Anna has $2 + 5 = 7$ apples. Elsa and Anna have $5 + 7 = 12$ apples together. The answer is 12.

Q: Elsa has 5 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 $2 + 5 = 7$ apples.
2. Elsa and Anna have $5 + 7 = 12$ apples together.

Q: {question}
A: Let’s break down this problem:

The answer is:

Source: Zhou et al. 2023
Reasoning in Large Language Models: prompting

Least-to-Most prompting for Math reasoning

<table>
<thead>
<tr>
<th>Accuracy by Steps (GSM8K)</th>
<th>All</th>
<th>2 Steps</th>
<th>3 Steps</th>
<th>4 steps</th>
<th>≥ 5 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least-to-Most</td>
<td>62.39</td>
<td>74.53</td>
<td>68.91</td>
<td>59.73</td>
<td>45.23</td>
</tr>
<tr>
<td>Chain-of-Thought</td>
<td>60.87</td>
<td>76.68</td>
<td>67.29</td>
<td>59.39</td>
<td>39.07</td>
</tr>
</tbody>
</table>

Generalizes to more #steps than in-context example!

Source: Zhou et al. 2023
Reasoning in Large Language Models: prompting

Least-to-Most prompting for Math reasoning

<table>
<thead>
<tr>
<th>Accuracy by Steps (GSM8K)</th>
<th>All</th>
<th>2 Steps</th>
<th>3 Steps</th>
<th>4 steps</th>
<th>≥ 5 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least-to-Most</td>
<td>62.39</td>
<td>74.53</td>
<td>68.91</td>
<td>59.73</td>
<td>45.23</td>
</tr>
<tr>
<td>Chain-of-Thought</td>
<td>60.87</td>
<td>76.68</td>
<td>67.29</td>
<td>59.39</td>
<td>39.07</td>
</tr>
</tbody>
</table>

Generalizes to more #steps than in-context example!

<table>
<thead>
<tr>
<th>Prompting method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero-Shot</td>
<td>16.38</td>
</tr>
<tr>
<td>Standard prompting</td>
<td>17.06³</td>
</tr>
<tr>
<td>Chain-of-Thought (original)</td>
<td>61.18</td>
</tr>
<tr>
<td>Chain-of-Thought (1-shot)</td>
<td>60.88</td>
</tr>
<tr>
<td>Least-to-Most (1-shot)</td>
<td>62.39</td>
</tr>
<tr>
<td>Chain-of-Thought (best)</td>
<td>68.61³</td>
</tr>
<tr>
<td>Least-to-Most (best)</td>
<td>68.01</td>
</tr>
</tbody>
</table>

But with enough prompt engineering, CoT ≈ Least-to-Most

Source: Zhou et al. 2023
Reasoning in Large Language Models via distillation

So far, we’ve only looked at prompting >100B parameter models for reasoning.

Can we get reasoning-like behavior with smaller LMs by teaching them to imitate larger models?
ORCA: Instruction-tuning small LMs with CoT Rationales

1. Collect a wide variety of instructions from the FLAN-v2 collection

<table>
<thead>
<tr>
<th>Mixture Name</th>
<th>Sampling Algorithm</th>
<th>Original Size</th>
<th>Sampled</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoT</td>
<td>No Sampling</td>
<td>150K</td>
<td>150K</td>
</tr>
<tr>
<td>NIV2</td>
<td>Stratified Sampling</td>
<td>5M</td>
<td>440K</td>
</tr>
<tr>
<td>FLAN2021</td>
<td>Stratified Sampling</td>
<td>&gt;28.9M</td>
<td>2.5M</td>
</tr>
<tr>
<td>T0</td>
<td>Stratified Sampling</td>
<td>85.7M</td>
<td>2M</td>
</tr>
<tr>
<td>Dialog</td>
<td>Skipped</td>
<td>22.5M</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Construction of our training data with 5 million samples.

Source: Mukherjee et al. 2023
ORCA: Instruction-tuning small LMs with CoT Rationales

1. Collect a wide variety of instructions from the FLAN-v2 collection.
2. Prompt GPT4 or ChatGPT with these instructions along with a system message

**System Instruction:** You are an AI assistant. User will give you a task. Your goal is to complete the task as faithfully as you can. While performing the task think step-by-step and justify your steps.

**User Instruction:** Use the given data to calculate the median.

**Input:** [7, 3, 8, 2, 10]

**Output:** To calculate the median, I will follow these steps:
Step 1: Arrange the data in ascending order [2, 3, 7, 8, 10]
Step 2: Determine if the data set has an odd or even number of values
The data set has 5 values, which is an odd number.
Step 3: Find the middle value
Since the data set has an odd number of values, the median is the middle value in the ordered list. The middle value in the list [2, 3, 7, 8, 10] is 7.

*Source: Mukherjee et al. 2023*
ORCA: Instruction-tuning small LMs with CoT Rationales

1. Collect a wide variety of instructions from the FLAN-v2 collection.
2. Prompt GPT4 or ChatGPT with these instructions along with a system message.
ORCA: Instruction-tuning small LMs with CoT Rationales

Evaluation: *BigBench-hard*, a collection of 23 Big-Bench tasks with a special focus on multi-step reasoning
ORCA: Instruction-tuning small LMs with CoT Rationales

Evaluation: BigBench-hard, a collection of 23 Big-Bench tasks with a special focus on multi-step reasoning

### Boolean Expressions

Q: True and False and not True and True is
A: Let's think step by step.
Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.
We first simplify this expression "Z" as follows: "Z = True and False and not True and True = A and B" where "A = True and False" and "B = not True and True".
Let's evaluate A: A = True and False = False.
Let's evaluate B: B = not True and True = not (True and True) = not (True) = False.
Plugging in A and B, we get: Z = A and B = False and False = False. So the answer is False.

Source: Suzgun et al. 2022
ORCA: Instruction-tuning small LMs with CoT Rationales

Evaluation: BigBench-hard, a collection of 23 Big-Bench tasks with a special focus on multi-step reasoning

Q: Tomorrow is 11/12/2019. What is the date one year ago from today in MM/DD/YYYY?

Options:
(A) 09/04/2018
(B) 11/11/2018
(C) 08/25/2018
(D) 11/02/2018
(E) 11/04/2018

A: Let’s think step by step.
If tomorrow is 11/12/2019, then today is 11/11/2019. The date one year ago from today is 11/11/2018. So the answer is (B).

Source: Suzgun et al. 2022
Evaluation: **BigBench-hard**, a collection of 23 Big-Bench tasks with a special focus on multi-step reasoning.

---

**Geometric Shapes**

**Q:** This SVG path element `<path d="M 14.19,26.04 L 51.43,39.21 L 58.44,36.69 L 56.63,30.17 L 48.53,26.66 L 14.19,26.04"/>` draws a

**Options:**

(A) circle  
(B) heptagon  
(C) hexagon  
(D) kite  
(E) line  
(F) octagon  
(G) pentagon  
(H) rectangle  
(I) sector  
(J) triangle

**A:** Let's think step by step.  
This SVG path element contains "M" and "L" commands. M takes two parameters (x,y) and moves the current point to the coordinates (x,y). L takes two parameters (x,y) and draws a line from the previous coordinate to the new coordinate (x,y).  
This path can be decomposed into 6 separate commands:

2. L 51.43,39.21: Create a line from 14.19,26.04 to 51.43,39.21.
3. L 58.44,36.69: Create a line from 51.43,39.21 to 58.44,36.69.
4. L 56.63,30.17: Create a line from 58.44,36.69 to 56.63,30.17.

This SVG path starts at point 14.19,26.04, creates five consecutive and touching lines, and then returns back its starting point, thereby creating a five-sided shape. It does not have any curves or arches. "pentagon" is the only five-sided polygon on the list. So the answer is **(G)**.

---

Source: Suzgun et al. 2022
## ORCA: Instruction-tuning small LMs with CoT Rationales

<table>
<thead>
<tr>
<th>Task</th>
<th>ChatGPT</th>
<th>GPT-4</th>
<th>Vicuna-13B</th>
<th>Orca-13B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean Expressions</td>
<td>82.8</td>
<td>77.6</td>
<td>40.8</td>
<td>72.0 (76.5%)</td>
</tr>
<tr>
<td>Causal Judgement</td>
<td>57.2</td>
<td>59.9</td>
<td>42.2</td>
<td>59.9 (41.8%)</td>
</tr>
<tr>
<td>Date Understanding</td>
<td>42.8</td>
<td>74.8</td>
<td>10.0</td>
<td>50.0 (400.0%)</td>
</tr>
<tr>
<td>Disambiguation QA</td>
<td>57.2</td>
<td>69.2</td>
<td>18.4</td>
<td>63.6 (245.7%)</td>
</tr>
<tr>
<td>Formal Fallacies</td>
<td>53.6</td>
<td>64.4</td>
<td>47.2</td>
<td>56.0 (18.0%)</td>
</tr>
<tr>
<td>Geometric Shapes</td>
<td>25.6</td>
<td>40.8</td>
<td>3.6</td>
<td>20.8 (477.8%)</td>
</tr>
<tr>
<td>Hyperbaton</td>
<td>69.2</td>
<td>62.8</td>
<td>44.0</td>
<td>64.0 (45.5%)</td>
</tr>
<tr>
<td>Logical Deduction (5 objects)</td>
<td>38.8</td>
<td>66.8</td>
<td>4.8</td>
<td>39.6 (725.0%)</td>
</tr>
<tr>
<td>Logical Deduction (7 objects)</td>
<td>39.6</td>
<td>66.0</td>
<td>1.2</td>
<td>36.0 (2900.0%)</td>
</tr>
<tr>
<td>Logical Deduction (3 objects)</td>
<td>60.4</td>
<td>94.0</td>
<td>16.8</td>
<td>57.6 (242.9%)</td>
</tr>
<tr>
<td>Movie Recommendation</td>
<td>55.4</td>
<td>79.5</td>
<td>43.4</td>
<td>78.3 (80.6%)</td>
</tr>
<tr>
<td>Navigate</td>
<td>55.6</td>
<td>68.8</td>
<td>46.4</td>
<td>57.6 (24.1%)</td>
</tr>
<tr>
<td>Penguins in a Table</td>
<td>45.9</td>
<td>76.7</td>
<td>15.1</td>
<td>42.5 (181.8%)</td>
</tr>
<tr>
<td>Reasoning about Colored Objects</td>
<td>47.6</td>
<td>84.8</td>
<td>12.0</td>
<td>48.4 (303.3%)</td>
</tr>
<tr>
<td>Ruin Names</td>
<td>56.0</td>
<td>89.1</td>
<td>15.7</td>
<td>39.5 (151.2%)</td>
</tr>
<tr>
<td>Salient Translation Error Detection</td>
<td>40.8</td>
<td>62.4</td>
<td>2.0</td>
<td>40.8 (1940.0%)</td>
</tr>
<tr>
<td>Snarks</td>
<td>59.0</td>
<td>87.6</td>
<td>28.1</td>
<td>62.4 (122.0%)</td>
</tr>
<tr>
<td>Sports Understanding</td>
<td>79.6</td>
<td>84.4</td>
<td>48.4</td>
<td>67.2 (38.8%)</td>
</tr>
<tr>
<td>Temporal Sequences</td>
<td>35.6</td>
<td>98.0</td>
<td>16.0</td>
<td>72.0 (350.0%)</td>
</tr>
<tr>
<td>Tracking Shuffled Objects (5 objects)</td>
<td>18.4</td>
<td>25.2</td>
<td>9.2</td>
<td>15.6 (69.6%)</td>
</tr>
<tr>
<td>Tracking Shuffled Objects (7 objects)</td>
<td>15.2</td>
<td>25.2</td>
<td>5.6</td>
<td>14.0 (150.0%)</td>
</tr>
<tr>
<td>Tracking Shuffled Objects (3 objects)</td>
<td>31.6</td>
<td>42.4</td>
<td>23.2</td>
<td>34.8 (50.0%)</td>
</tr>
<tr>
<td>Web of Lies</td>
<td>56.0</td>
<td>49.6</td>
<td>41.2</td>
<td>51.2 (24.3%)</td>
</tr>
</tbody>
</table>

- Outperforms Vicuna-13B
- Outperforms ChatGPT!
- GPT-4 has potential data contamination issues with Bigbench-hard

Source: Mukherjee et al. 2023
Reasoning by Finetuning LMs on their own outputs?

ReST\textsuperscript{EM} alternates between the following two steps:

1. Generate (E-Step): Given reasoning problem, sample multiple solutions from language model. Filter based on some (problem specific) function [answer correctness for math problems]

2. Improve (M-Step): Update the language model to maximize probability of filtered solutions, using supervised finetuning

Source: Singh et al. 2024
Reasoning by Finetuning LMs on their own outputs?

ReST\textsuperscript{EM} alternates between the following two steps:

1. Generate (E-Step): Given reasoning problem, sample multiple solutions from language model. Filter based on some (problem specific) function \([\text{answer correctness for math problems}]\)

2. Improve (M-Step): Update the language model to maximize probability of filtered solutions, using supervised finetuning

Source: Singh et al. 2024
Can Language Models Reason?

Large language models' ability to generate text also lets them plan and reason

TechTalks
Large language models have a reasoning problem

Microsoft Says New A.I. Shows Signs of Human Reasoning

Some Glimpse AGI in ChatGPT. Others Call It a Mirage

Source: https://aiguide.substack.com/p/can-large-language-models-reason
Can Language Models Reason?

CoT Rationales are often not faithful

Source: Lanham et al. 2023
Can Language Models Reason?
Reasoning vs Memorization: Using Counterfactuals

Source: Wu et al., 2024
## Can Language Models Reason?

### Reasoning vs Memorization: Using Counterfactuals

#### Original transformation types

<table>
<thead>
<tr>
<th>Extend sequence</th>
<th>Successor</th>
<th>Predecessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd</td>
<td>abcde</td>
<td>bcde</td>
</tr>
<tr>
<td>ijkI</td>
<td>iklm</td>
<td>ikl</td>
</tr>
<tr>
<td>Remove redundant letter</td>
<td>Fix alphabetic sequence</td>
<td>Sort</td>
</tr>
<tr>
<td>abcdde</td>
<td>abcde</td>
<td>adcbde</td>
</tr>
<tr>
<td>ijkklm</td>
<td>ijklm</td>
<td>kjml</td>
</tr>
</tbody>
</table>

#### Modified transformation types

<table>
<thead>
<tr>
<th>Extend sequence</th>
<th>Successor</th>
<th>Predecessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd</td>
<td>abcde</td>
<td>cdef</td>
</tr>
<tr>
<td>ijkI</td>
<td>ikln</td>
<td>jklm</td>
</tr>
<tr>
<td>Remove redundant letter</td>
<td>Fix alphabetic sequence</td>
<td>Sort</td>
</tr>
<tr>
<td>acegi</td>
<td>acegi</td>
<td>kfpau</td>
</tr>
<tr>
<td>ikkmoq</td>
<td>ikmoq</td>
<td>imkoq</td>
</tr>
</tbody>
</table>

Source: Hodel et al. 2024
Can Language Models Reason?

Reasoning vs Memorization: Using Counterfactuals

Source: Hodel et al. 2024
Language Model Agents

with some slides borrowed from Frank Xu (CMU)
Some Terminology

Environment

Observation

Action

π(· | g)

“Instruction-following agent”

“Language conditioned policy”

“digital agent”

Language Instruction
Some Terminology

Type … on …, Click on …, Choose … from dropdown, …

Environment

Observation

$\pi(\cdot | g)$

“Book a flight from San Francisco to New York”

Raw pixels as observation?  HTML DOM as observation?
Applications: Natural Language Interfaces

Virtual Assistants
- Set an alarm at 7 AM
- Remind me for the meeting at 5pm
- Play Jay Chou’s latest album

Natural Language Programming
- Sort my_list in descending order
- Copy my_file to home folder
- Dump my_dict as a csv file output.csv
Applications: UI automation

Click the "Menu" button, and then find and click on the item with the ▶ icon.

“Play some synthwave songs”
Applications: Multi-step “Tool use”
Instruction following agents [Pre LLMs]

Idea #1: Directly map from instructions to action sequences like Machine Translation [works well for simple grounded environments like text2sql, knowledge graph querying]

\[
\max_\theta p_\theta(\{a_1, a_2, \ldots\} \mid g)
\]

**a)** What states border Texas
\[
\lambda x.\text{state}(x) \land \text{borders}(x, \text{texas})
\]

**b)** What is the largest state
\[
\arg \max(\lambda x.\text{state}(x), \lambda x.\text{size}(x))
\]

**c)** What states border the state that borders the most states
\[
\lambda x.\text{state}(x) \land \text{borders}(x, \arg \max(\lambda y.\text{state}(y), \\
\lambda y.\text{count}(\lambda z.\text{state}(z) \land \text{borders}(y, z))))
\]

Source: Zettlemoyer et al. 2012
Instruction following agents [Pre LLMs]

Idea #2: Infer executable, latent plans from (instruction, trajectory) pairs and train a model to go from instructions to plans

Instruction: “Place your back against the wall of the ‘T’ intersection. Turn left. Go forward along the pink-flowered carpet hall two segments to the intersection with the brick hall. This intersection contains a hatrack. Turn left. Go forward three segments to an intersection with a bare concrete hall, passing a lamp. This is Position 5.”

Parse: Turn (), Verify ( back: WALL ), Turn ( LEFT ), Travel ( ), Verify ( side: BRICK HALLWAY ), Turn ( LEFT ), Travel ( steps: 3 ), Verify ( side: CONCRETE HALLWAY )

Source: Chen et al. 2011
Instruction following agents [Pre LLMs]

Idea #3: Use RL to directly map instructions to actions

$$\max_{\theta} \mathbb{E}_{a \sim \pi_\theta} R(a; \text{instruction, observation})$$

Source: Branavan et al. 2009
Instruction following agents [in 2024]

Main Idea: Use causal transformers for \textit{generative trajectory modeling!}

\[
p(\tau \mid g) = p(s_1, a_1, s_2, a_2, \ldots \mid g) = \prod_t p(s_t \mid s_{t-1}, a_t) \times \pi(a_t \mid \tau_{\leq t}, g)
\]
A Simple Language Model Agent with ReACT

You are an agent capable of the following actions:
1. Type X on Y
2. Move mouse to
3. Click on X
4. Type Char x on Y

Your objective is to follow user instructions, by mapping them into a sequence of actions.
Instruction: \{g\}

So far, you have taken the following actions and observed the following environment states:

Previous Actions and Observations:
o1:
a1:
o2:
a2:
...

After executing these actions, you observe the following HTML state: <HTML state>

Now, think about your next action:
Thought: [model-pred]

Now, take an action:
Action: [model-pred]
Some popular benchmarks for LM agents:

**MiniWoB++**

Sandboxed environment evaluating basic browser interactions across a range of applications from social media to email clients

Evaluates functional correctness

Not real world (limited functionality)

Relatively short-horizon

Zero-shot performance far from perfect!

Source: Shi et al. 2017
Some popular benchmarks for LM agents: WebArena

“Create a plan to visit Pittsburgh’s art museums with minimal driving distance starting from Schenley Park. Log the order in my “awesome-northeast-us-travel” repository”

Environment with sandboxed approximations of real websites spanning e-commerce, social media!

Additional utility tools: Maps, calculators, scratchpads, Wikipedia...

Multi-tab browsing

Long-horizon tasks

Evaluates functional correctness

Source: Zhou et al. 2024
Some popular benchmarks for LM agents:

**WebLINX**

- Web-interactions on real websites
- Conversational: includes a new “say” action to communicate with human to gather information
- Multi-tab browsing
- Turn-level metrics for evaluation
- Not an environment, but a collection of interactions

Source: Lü et al. 2024
Training data for Language Model Agents

- Standard practice: In-context learning with few-shot demonstrations of humans performing following similar instructions.
- This is still not scalable / reliable

1000s of environments, many kinds of interactions possible...

Can agents autonomously explore their environments to construct high quality synthetic demonstrations?
Use Exploration + Model Generated Data!

Prompt: Given a website, take actions of the following format to explore...

Action: \([\text{pred}]\)
Use Exploration + Model Generated Data!

How can we decide if a sequence of interactions is meaningful? *Use Natural Language!*

Prompt: You are given a sequence of actions and corresponding HTML states on a website...

Label: \[[pred]\]
Use Exploration + Model Generated Data!

“Book a flight from SFO to NYC”

How can we decide if a sequence of interactions is meaningful? *Use Natural Language!*
How can we decide if a sequence of interactions is meaningful? **Use Natural Language!**
Use Exploration + Model Generated Data!

Instruction: Set the date as 12/26/2016

Trajectory

Prompt: Map the given instruction to a sequence of actions, one at a time.
Thought: [[pred]]
Action: [[pred]]

\[ \pi_{LM}(\cdot \mid g) \]

Instruction: Set the date as 12/26/2016

\[ R(g, \tau) \]

Prompt: Output "1" if the trajectory is correct for the given instruction...
Label: [[pred]]
Use Exploration + Model Generated Data!

Instruction: Set the date as 12/26/2016

$\pi_{LM}(\cdot \mid g)$

$R(g, \tau)$

Trajectory
Use Exploration + Model Generated Data!

Instruction: Book a flight from SFO to NYC

Set origin to SFO and dest to NYC
Use Exploration + Model Generated Data!

Instruction: Book a flight from SFO to NYC

Trajectory

Set origin to SFO and dest to NYC

$\pi_{LM}(\cdot | g)$

$p_{label}(\cdot | \tau)$

$R(g, \tau)$
BAGEL: Use Exploration + Model Generated Data!

Prompt: Map the given instruction to a sequence of actions, one at a time.
Thought: [[pred]]
Action: [[pred]]

Prompt: You are given a sequence of actions and corresponding HTML states on a website...
Label: [[pred]]

πLM(· | g)  →  p_label(· | τ)

BAGEL

(Bootstrapping Agents by Guiding Exploration with Language)
1. Explore Environment to collect trajectories

2. Create Synthetic demonstrations via iterative re-labeling

3. Instruction-Following (Inference Time): Retrieve Relevant Demonstration via retrieval to use as in-context exemplars

"Book the cheapest flight from Denver to LA"

Book the cheapest … ➔ {type on …, select …, click …}
Buy a flight from Denver … ➔ {type …, click …, select …}

Finetuning possible too!
Multimodality?

- So far, we’ve looked at using text-only language models for agents
- This is intractable for real-world UIs with very long HTML
- Can we instead operate directly over pixel space?
Multimodality

LLaVA
Multimodality

Pix2Struct

Transformer Encoder

Patch + Position Embedding
* Extra learnable [class] embedding

Linear Projection of Flattened Patches

Transformer Decoder

Programming Survey
Your favorite programming language:
- Python
- C++
- Java
Submit

Programming Survey
Your favorite programming language:
- C++
- Java
Submit

<<Python>>
<img src=py_logo.png img_alt=Python>>
<<C++>>
<img src=cpp_logo.png img_alt=C++>>
<<Java>>
<img src=java_logo.png img_alt=Java>>
<Submit>
LM Agents is an emerging application!

The “prompting gap”: without extensive prompting / bespoke few-shot examples, competitive LMs are far from perfect on even the simplest environments
LM Agents is an emerging application!

Long-horizon planning is hard: Even on simple benchmarks, performance drops drastically on tasks that require longer horizon planning.
LM Agents is an emerging application!

Latest Work: BrowserGym 25%
More prompt engineering
More observation/action interface engineering
LM Agents is an emerging application!

**S2:** Open Google translate and sign in using the following credentials: [email] [password]

**Reference (B):** [password]  
**GPT-4V (R):** [email]  
**LLaMA (B):** [password]