Exploring Reasoning and Interactive Benchmarking of Language Models

Akshara Prabhanar

Princeton University

April 2024
Viewing Chain-of-Thought as a Probabilistic, Memorization-Influenced Noisy Reasoning Process
Reasoning x LLM Landscape

What’s reasoning in LLMs?
- no clear definition
- focus on informal deductive reasoning

Huang and Chang et. al 2022

CoT elicits “reasoning” of LLMs

Wei et. al 2022

Are LLMs truly reasoning?

PlanBench
Shaikh et. al 2023

Human-like content effects displayed
Dasgupta et. al 2022

Solving BBH tasks at par with humans
Suzgun et. al 2022

Synthetic QA tasks
Saparov and He 2022

Biased by freq of certain “terms”
Razhegi et. al 2022

Generalization vs. Memorization:
Do LLMs exhibit true reasoning or rely on heuristics (memorization)?

inspired from Huang and Chang et. al, Wei et. al 2022
Formulation

Task: Decoding Shift ciphers

Data

- 7 letter output words
- controlled for tokens
- 5 probability bins (GPT-2)

Test Model

Independent manipulations

- difficulty
- frequency
- answer probability
- reasoning steps

Rot-13 is a cipher in which each letter is shifted 13 positions forward in the alphabet. For example, here is a message written in rot-13:

Rot-13 text: “fgnl”

To decode this message, we shift each letter 13 positions backward:
1. f → s
2. g → t
3. n → a
4. l → y

Therefore, the original text is: “stay”

Here is another message in rot-13. Decode this message one letter at a time. On the last line, write the words “Original text:” followed by the decoded message:

Rot-13 text: <test_input>

Figure 5: Text-based CoT prompt consisting of a description and the demonstration that includes several reasoning steps.
Results

Hypothetical accuracy of different types of reasoning

Actual accuracy at step-level and overall decoding

Logistic regression analysis

\[
\text{correct} \sim \min(\text{shift\_level, 26-shift\_level}) + \text{input\_logprob} + \text{output\_logprob} + \text{shift\_freq}
\]

Significant features: shift\_level, output\_logprob, shift\_freq
Results

Isolating reasoning

Math only CoT – nearly perfect system

Math only CoT prompt

Rot-13 is a cipher in which each letter is shifted 13 positions forward in the alphabet. For example, here is a message written in rot-13:
Rot-13 text: "fgnl"

To decode this message, we need to shift each letter 13 positions backward. Let's start by writing the position-letter mapping for the alphabet:

a -> 0
b -> 1
c -> 2
...

Next, we find the encoded letter as follows:
Position of original letter = (Position of given letter - 13) mod 26

Shift-13 is a process in which each number is shifted 13 positions forward until it reaches 26 and subsequently circles back to 1. For example, here is a sequence of numbers written in shift-13:
shift-13 sequence: "6,7,14,12"

To decode this sequence, we need to shift each number 13 positions backward.
New position = (Given position - 13) mod 26

Using this,
1. 5 -> (5 - 13) mod 26 -> 18
2. 6 -> (6 - 13) mod 26 -> 19
3. 13 -> (13 - 13) mod 26 -> 0
4. 11 -> (11 - 13) mod 26 -> 24

Therefore, the original sequence of numbers is: "19,20,1,25"

Here is another sequence of numbers in shift-13. Decode this sequence one number at a time. On the last line, write the words "Original sequence: " followed by the decoded sequence:
shift-13 sequence: <encoded_test_input>

Math CoT – no implicit memorization of a–z
Results

#1 Probabilistic effects

<table>
<thead>
<tr>
<th>Prob</th>
<th>Chain Steps Output</th>
<th>Final Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>High</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>40</td>
</tr>
<tr>
<td>Medium</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>58</td>
</tr>
<tr>
<td>Low</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>78</td>
</tr>
</tbody>
</table>

(a) rot-4.

positive influence for high probability bins,
negative influence for low probability bins

#2 Memorization effects

<table>
<thead>
<tr>
<th>Prob</th>
<th>Chain Steps Output</th>
<th>Final Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>High</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>29</td>
</tr>
<tr>
<td>Medium</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>46</td>
</tr>
<tr>
<td>Low</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>65</td>
</tr>
</tbody>
</table>

(b) rot-13.

incorrect step outputs leading to correct final answers 2x times on average in rot-13

#3 Noisy reasoning

appearance of peaks at 26 - shift_level in Math-CoT and Text-CoT.

<table>
<thead>
<tr>
<th>Style / Shift Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text CoT</td>
<td>49.2</td>
<td>52.4</td>
<td>50.8</td>
<td>51.6</td>
<td>42.8</td>
<td>47.0</td>
<td>32.6</td>
<td>44.0</td>
<td>30.2</td>
<td>9.6</td>
<td>0.2</td>
<td>0.0</td>
<td>60.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>16.6</td>
<td>17.0</td>
<td>17.0</td>
<td>47.0</td>
<td>45.4</td>
<td>50.4</td>
</tr>
<tr>
<td>Math CoT</td>
<td>70.2</td>
<td>68.0</td>
<td>68.6</td>
<td>66.2</td>
<td>58.8</td>
<td>60.4</td>
<td>57.4</td>
<td>63.2</td>
<td>45.4</td>
<td>41.6</td>
<td>40.2</td>
<td>33.8</td>
<td>71.0</td>
<td>16.2</td>
<td>19.2</td>
<td>17.8</td>
<td>11.4</td>
<td>8.8</td>
<td>4.8</td>
<td>1.6</td>
<td>0.8</td>
<td>0.6</td>
<td>0.0</td>
<td>50.0</td>
<td>61.6</td>
</tr>
</tbody>
</table>

Table 2: Accuracy % comparison between Text-based and Math-based CoT prompt styles. Implicit need for memorization in text-based CoT causes an overall reduced performance. Model performs best at shift levels 13, 1, 3, and 2 which are the most frequently occurring shifts in real-world corpora highlighting the memorization effects.
Results

Self-conditioning idea

- validity of demonstration
- relevant intermediate step outputs

❓ Generalization vs. Memorization

Middle ground – clear hallmarks of being influenced by “memorization”, but also shows clear hallmarks of what would be expected by “true reasoning”
InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback
Code x LLM Landscape

Why Code?
- Reasoning
- Well Structured
- Practical Applications

Evaluating Code Generation
Sequence to Sequence, Transduction Problems

What are the names of the states where at least 3 heads were born?

SELECT born_state FROM head GROUP BY born_state HAVING count(*) >= 3

Popular Benchmarks
HumanEval (.py), MBPP (.py), APPS (.py), Spider (.sql), NL2Bash (.sh)
Evaluation Methods

How do we evaluate generations?

Semantic Similarity

What about execution oriented evaluation?

Paper-specific task environments (i.e. compiler, feedback signals, execution procedure) make it difficult to compare existing methods.

A benchmark is needed to **define, standardize, and enable easy construction of interactive coding tasks.**
InterCode

Lightweight, flexible, and easy-to-use framework for designing interactive coding tasks to evaluate language agents that can code.

Contributions
• Define formulation for “Interactive Coding Task”
• Release a Python library to enable task creation
• Create 3 task environments: Bash, SQL, Python
• Run baseline experiments
Interactive Coding Task

Standard RL formulation (POMDP), where an action is code, and observation is execution feedback.
InterCode Library

The InterCode source code and pypi package enables practitioners to build interactive coding tasks in 3 steps.

1. **Dockerfiles**, to define safe, customizable, reproducible envs
2. **Existing or new datasets** consisting of [Inst., Gold] Pairs.
3. **Reward Function** based on gold and agent execution diff.
**InterCode-SQL**

**Task Setting**

**Setting:** MySQL Database  
**Dataset:** Spider  
**Action Space:** SQL commands

**Example Task Instance:**

**Instruction:** Find the first name and age of students who have a dog but not a cat.

**Gold:**
```
SELECT T1.fname, T1.age FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON...
```

(5 more lines)
**InterCode–SQL**

**Submission:** Std. Out

**Reward Function:**

Intersection over Union score that accounts for duplicates and order

\[
R = \frac{A \cap G}{A \cup G} \times \frac{kendalltau((A \cap (A \cap G)), (G \cap (A \cap G))) + 1}{2}
\]
Experiments

We evaluate state-of-the-art language models, equipped with 1 of 4 different reasoning frameworks:

- **Single Turn**: No interaction
- **Try Again**: Multi-turn interaction with execution feedback
- **Plan & Solve**: Write a plan, then execute it across multiple turns
- **ReAct**: Interleaving reasoning and acting

Metrics

- **Success Rate**: # of Tasks w/ Reward of 1
- **# of Turns**
- **Error %**: Rate of admissible actions
Results

Takeaway #1: Interaction helps models perform better on coding tasks

From comparing Single Turn and Try Again mode, we note that performance on task of generating correct answer to an instruction using code is vastly superior in an interactive setting.
Takeaway #2: Different task envs test different skills

InterCode–SQL requires more context discovery and error correction. InterCode–Bash requires planning and modularizing tasks into several steps.

Action 1: SHOW TABLES
Action 2: DESC <Table>
Action 3: SELECT _ FROM <Table>...
Action 4: SELECT _ FROM <Table>...

Plan: 1. Add prefix to all files in directory 2. Move into separate folder
Action 1: rename -n -A <prefix>- *
Action 2: mv -n -A <prefix>- *

From comparing different strategies (ReAct, Plan & Solve), we observe that

A. Different tasks present different learning challenges, and
B. More adaptive reasoning techniques are favorable.
Results

Takeaway #3: Evaluating Cybersecurity Skills with Capture the Flag

❓ Can language models exploit security vulnerabilities with code?
🔧 Use InterCode to create Capture the Flag task

GPT-4 struggles with adjusting its problem solving approach when it hits a dead end (1st 3 to 4 turns).
Future directions

- Understanding other reasoning methods, evaluating open-source models (scale)
- Automatic ways of task collection and extending environment to support multi-modal settings
Thank You!

Shunyu Yao, Prof. R. Thomas McCoy, Prof. Karthik Narasimhan, Prof. Thomas L. Griffiths