Deciphering the Factors Influencing the Efficacy of Chain-of-Thought: Probability, Memorization, and Noisy Reasoning

Akshara Prabhakar,[1] Thomas L. Griffiths,[1,2] R. Thomas McCoy[3,4] [*]

[1]Department of Computer Science, Princeton University
[2]Department of Psychology, Princeton University
[3]Department of Linguistics, Yale University
[4]Wu Tsai Institute, Yale University

Correspondence: akshblr555@gmail.com

Abstract

Chain-of-Thought (CoT) prompting has been shown to enhance the multi-step reasoning capabilities of Large Language Models (LLMs). However, debates persist about whether LLMs exhibit *abstract generalization* or rely on *shallow heuristics* when given CoT prompts. To understand the factors influencing CoT reasoning we provide a detailed case study of the symbolic reasoning task of decoding shift ciphers (Andress, 2014), where letters are shifted forward some number of steps in the alphabet. We analyze the pattern of results produced by three LLMs—GPT-4, Claude 3, and Llama 3.1—performing this task using CoT prompting. By focusing on a single relatively simple task, we are able to identify three factors that systematically affect CoT performance: the probability of the task's expected output (probability), what the model has implicitly learned during pre-training (memorization), and the number of intermediate operations involved in reasoning (noisy reasoning). We show that these factors can drastically influence task accuracy across all three LLMs; e.g., when tested with GPT-4, varying the output's probability of occurrence shifts accuracy from 26% to 70%. Overall, we conclude that CoT prompting performance reflects both memorization and a probabilistic version of genuine reasoning.[†]

1 Introduction

Reasoning, one of the key aspects of human intelligence, is the process of thinking about something logically and systematically using evidence and past experiences to make a decision (Wason, 1968; Wason and Johnson-Laird, 1972; Fagin et al., 2004). The impressive performance of Large Language Models (LLMs) across a wide range of tasks has spurred extensive research into their reasoning capabilities (Huang and Chang, 2023; Qiao et al., 2023). It remains unclear whether the behavior of these systems is based on true reasoning or on shallow heuristics. Some results provide evidence that LLMs are able to reason (Suzgun et al., 2023; Lampinen et al., 2024; Saparov and He, 2023), while others show that they still struggle on tasks that humans can easily solve via reasoning (Han et al., 2022; Valmeekam et al., 2023; McCoy et al., 2023; Razeghi et al., 2022; Cao et al., 2024).

The Chain-of-Thought (CoT; Wei et al., 2022) prompting strategy has played a significant role in this debate. CoT involves prompting an LLM to generate a sequence of intermediate reasoning steps before producing the final answer, given some in-context exemplar(s) of how to break the task into steps. CoT and its several variants (Kojima et al., 2022; Zhou et al., 2023; Wang et al., 2023b) have been shown to substantially improve performance over standard prompting. Recent works have tried to identify which aspects of the demonstration contribute to CoT's enhanced performance (Huang and Chang, 2023; Madaan and Yazdanbakhsh, 2022; Jin et al., 2024), typically relying on assessing performance across a wide range of tasks.

In this work, we take a different approach: we present an extensive case study on a single task that allows us to disentangle reasoning from memorization. The task we selected is solving shift ciphers, a simple type of code in which each letter is shifted forward a certain number of positions in the alphabet (Figure 1, panel 1). We choose this task because it allows us to independently manipulate several factors that could be relevant for characterizing how LLMs solve reasoning tasks when prompted with CoT: difficulty, frequency, and answer probability.

Our results suggest that CoT performance reflects three factors: probability, memorization, and noisy reasoning. First, the accuracy of CoT is affected by the probability of the correct output, with more probable outputs resulting in a stronger effect of CoT. Second, performance is higher when mem-

[*]Work begun while at Princeton University.

[†]Code and data are available at `https://github.com/aksh555/deciphering_cot`.

Findings of the Association for Computational Linguistics: EACL 2024, pages 3710–3724
November 12-16, 2024 ©2024 Association for Computational Linguistics

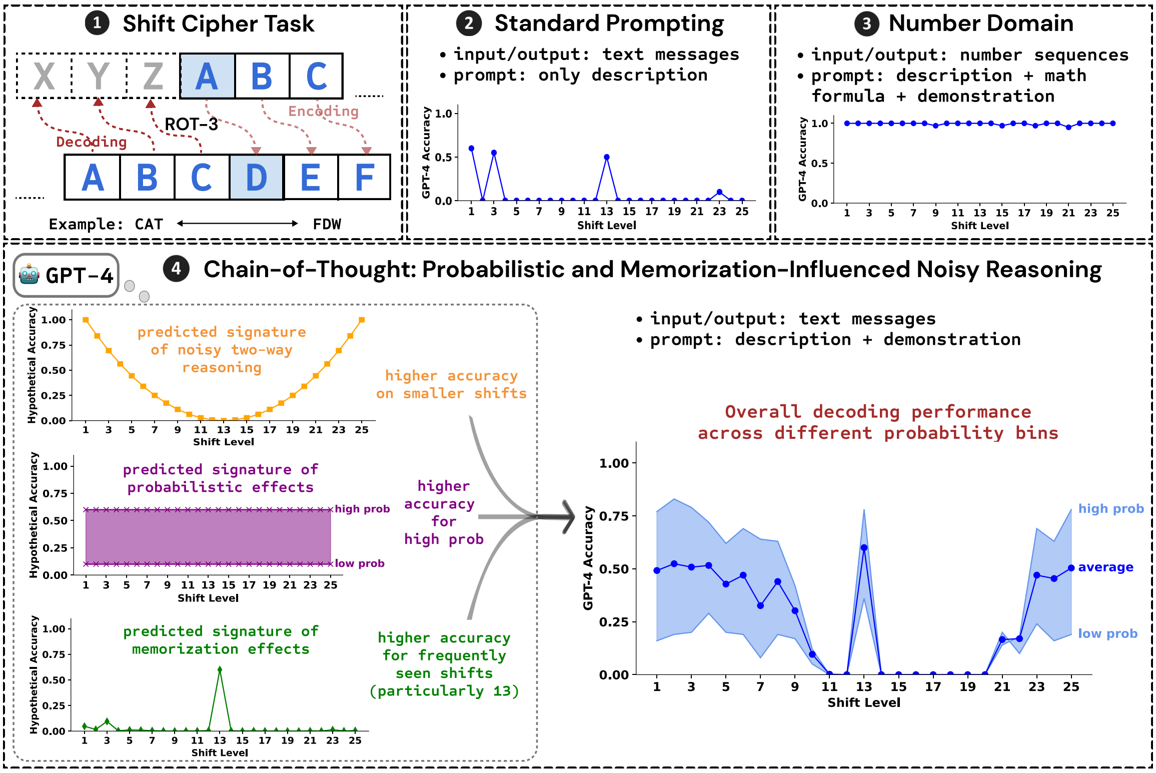


Figure 1: Overview. (1) Task: We have LLMs decode messages written in a shift cipher, in which each letter is shifted a fixed number of positions forward in the alphabet. (2) With standard prompting, GPT-4 performs poorly across most shift levels. (3) However, GPT-4 scores nearly perfectly on an isomorphic task based on numbers rather than letters. (4) With CoT prompting, GPT-4 adopts probabilistic and memorization-influenced noisy reasoning. That is, its performance (right) combines the trends we have hypothesized for each of the three factors on the left.

orization is possible, as indicated by the frequency of encountering different shift cipher variants during pre-training. The effects of probability and memorization show that CoT performance is not fully systematic abstract reasoning. Nonetheless, CoT performance is not solely driven by superficial heuristics: it also shows some hallmarks of true reasoning—albeit a noisy version of true reasoning, in which the error rate increases along with task difficulty (where we quantify task difficulty by the number of implicit reasoning steps involved). In addition, we find evidence that the effect of CoT fundamentally depends on generating sequences of words that increase the probability of the correct answer when conditioned upon; as long as this is the case, CoT can thus succeed even when the demonstrations in the prompt are invalid. In the ongoing debate about whether LLMs reason or memorize (Feldman, 2020; Zhang et al., 2023a; Magar and Schwartz, 2022; Srivastava et al., 2024; Antoniades et al., 2024), our results thus support a reasonable middle-ground: *LLM behavior displays aspects of both memorization and reasoning, and also reflects the probabilistic origins of these models.*

2 Related Work

In-Context Learning in Language Models. It has been argued that LLMs can *learn* a task purely from demonstrations given in their context without any additional training (Brown et al., 2020), a phenomenon known as in-context learning (ICL). There have been many investigations into how ICL operates. Theoretical frameworks have modeled the pretraining data as a mixture of Hidden Markov Models (Xie et al., 2022) and have argued that ICL is the result of implicit Bayesian inference (Zhang et al., 2023b), an argument supported with evidence from synthetic data and tasks (Chiang and Yogatama, 2024). The emergence of ICL has been attributed to factors including data distributional properties (Chan et al., 2022), pretraining term frequencies (Razeghi et al., 2022), and the creation of task vectors (Hendel et al., 2023). However, the extent to which ICL is *true learning* is unclear (Pan et al., 2023; Min et al., 2022). For example, Kossen et al. (2024) shows that ICL relies on in-context label information but cannot fully overcome preferences acquired during pre-training, which is evidence against the view that ICL is true learning.

Understanding CoT. Theoretical arguments have been formulated about how CoT improves the expressivity (Feng et al., 2023; Li et al., 2024; Merrill and Sabharwal, 2024) and sample complexity (Li et al., 2023) of ICL. Empirical studies have shown that CoT performance can be dramatically influenced by many features of the CoT prompt (Madaan and Yazdanbakhsh, 2022; Wu et al., 2023; Jin et al., 2024); for example, the relevance and ordering of reasoning fragments is more important than their accuracy (Wang et al., 2023a,b; Ye and Durrett, 2022), and minor input perturbations can substantially bias models' answers (Turpin et al., 2024), suggesting that the models lack general reasoning abilities (Stechly et al., 2024).

Our high-level goal in this work is to characterize what *type* of reasoning happens in LLMs when they are prompted with CoT prompting: to what extent is CoT performance driven by abstract reasoning vs. simple heuristics such as memorization? This question also implicitly underlies many of the papers discussed above, but directly focusing on this question leads us to investigate probability, frequency, and difficulty (see Section 4)—a different set of factors than those studied in prior work.

3 Approach

One challenge of evaluating the role of memorization and reasoning in the performance of LLMs is that these models are typically evaluated on a wide range of complex reasoning tasks, whose variety and complexity can obscure the factors that drive performance. By contrast, we propose to tease apart the factors behind the efficacy of CoT prompting by focusing on a single relatively simple task: deciphering text encoded with a shift cipher.

Encoding a message with a shift cipher involves replacing every letter with another letter that is some fixed number of positions (called $shift\_level$) forward in the alphabet; decoding is the reverse (shifting backward) as shown in Figure 1. These are also known as rotation ciphers since they rotate the alphabet forward some number of steps, and they are given the name `rot-`k where k corresponds to $shift\_level$. For example, given the test word "FDW" and that `rot-3` encryption has been used ($shift\_level = 3$), decoding involves shifting every letter 3 steps backward—i.e., $F \rightarrow C$, $D \rightarrow A$, and $W \rightarrow T$ to obtain "CAT" as the output. In our experiments, we give an LLM a single word encoded with a shift cipher and ask it to decode this text to recover the original word.

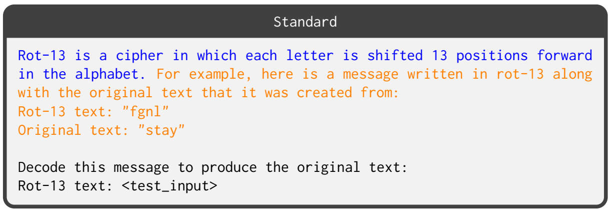
3.1 Motivation for using shift ciphers

Our main reason for using shift ciphers is because they involve a sharp dissociation between task complexity and task frequency (a key factor in memorization). The complexity of the decipherment task is determined by the shift level—ciphers that require more intermediate steps are more complex. Different shift levels also vary in their frequency in internet text (McCoy et al., 2023), and hence in the training data of large language models. Specifically, `rot-13` is widely used in internet forums to conceal text such as puzzle solutions and spoilers, and `rot-3` and `rot-1` commonly appear in tutorials on decipherment (`rot-3` is also known as the Caesar cipher, having apparently been used by the eponymous Caesar to encrypt his messages). In addition, shift ciphers facilitate investigation of the effect of probability because the correct answer can be any string, allowing us to modulate the probability of that string easily. Further, the systematic nature of the task makes it easy to generate examples and to verify correctness. Finally, decoding each letter in the message is an independent step, allowing us to easily analyze these individual steps.

McCoy et al. (2023) previously evaluated GPT models on shift ciphers, focusing on standard prompting along with some initial results using CoT. They study the effect of only probability and memorization, while we conduct a more extensive investigation into LLM behavior when prompted with CoT by additionally studying the influence of complexity and analyzing more models. Importantly, we add nuance to their findings by arguing for a middle-ground viewpoint that acknowledges the LLM weaknesses identified by McCoy et al. (2023) but also brings in novel observations that highlight the hallmarks of true reasoning that are present in these systems.

3.2 The effect of CoT on shift ciphers

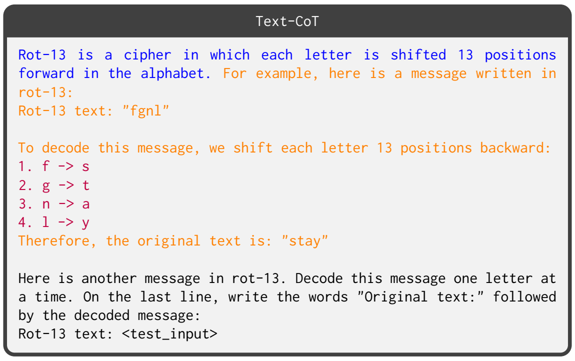
Data. We constructed a dataset comprising 7-letter words having exactly 2 tokens (measured using the tokenizer used by GPT-4) to control for confounding factors relating to tokenization. We found all 3-letter and 4-letter tokens from the lowercase English alphabet and formed words by considering possible combinations of 3-letter word-initial tokens followed by 4-letter non-word-initial tokens.



```
Standard
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 along
with the original text that it was created from:
Rot-13 text: "fgnl"
Original text: "stay"

Decode this message to produce the original text:
Rot-13 text: <test_input>
```

Figure 2: Standard prompt having just the description and demonstration.



```
Text-CoT
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 3: Text-CoT prompt consisting of a description and demonstration that includes several reasoning steps.

Following McCoy et al. (2023), we compute the log probability as the log probability that GPT-2 (Radford et al., 2019) assigns to the sentence ‘The word is "WORD"’, minus the log probability that it assigns to ‘The word is "’; thus, this yields the log probability assigned to just the word and the following quotation mark in the context of ‘The word is "’. The closing quotation mark is included because it indicates the end of the word. The words were scored by their log probability and arranged in descending order. Subsequently, five bins were formed by selecting equidistant log probability values as centers, with bin1 having the highest probability and bin5 having the lowest probability. We manually checked the words in this dataset and filtered them to ensure there were no inappropriate words used to obtain 150 words for each bin. We partitioned the 150 examples into two subsets: a subset containing 100 words used to evaluate GPT-4, and a subset containing 50 words used to evaluate logistic regression models that were fitted to GPT-4's performance on the 100-word subset. We prepared the inputs for the models by producing the shift-cipher-encoded versions of the words from the 5 probability bins across 25 shift levels (1 to 25). We ran all evaluations a single time; the accuracies that we report are accuracies over these 100-example sets.

We then assessed performance on this dataset using a variety of different prompts:

– **Standard.** This is a prompt with just the description of the task and demonstration but no reasoning steps (Figure 2).
– **Text-CoT.** This prompt encourages the model to decode a message one letter at a time (Figure 3). We chose this way of framing the CoT prompt following McCoy et al. (2023), who tried several variants and found this to be the best. To get a reasoning step correct, the model must have learned the alphabet during pre-training.
– **Math-CoT.** The prompt (Appendix A.1 Figure 9) encourages a reasoning pipeline that involves translating each letter to a number, performing the shift by applying arithmetic to this number, then converting the result back to a letter. The prompt also specifies the mapping between letters and positions, eliminating the need for the model to have internalized the positions of the letters in the alphabet.
– **Number-sequence CoT (Number-CoT).** This prompt (Appendix A.1 Figure 10) makes use of an alternative task that is isomorphic to shift ciphers but based in the number domain—the input and output are number sequences instead of letter sequences. Reasoning involves applying arithmetic to the input elements in the number sequence to get a corresponding output sequence.

We ran experiments using both open and closed source models: GPT-4 (gpt-4-0613) (OpenAI, 2023), Claude 3 (claude-3-opus-20240229) (Anthropic, 2024), and Llama-3.1-405B-Instruct (MetaAI, 2024). The reason for using such strong models is that their shift cipher performance is significantly improved by prompting with chain of thought. Additionally, this helps us to control several sources of extraneous errors making it easier to focus on the task itself and isolate the factors affecting CoT: It ensures that the format of the demonstration is closely followed, and that copy errors (errors in copying information from the prompt such as letters from encoded text and letter-position mappings) are rare. We set temperature to 0 and max_new_tokens to 200.

Figure 1 provides some initial results for GPT-4. Using standard prompts, GPT-4 gets zero accuracy across most shift levels, but it improves substantially (to an average accuracy of 32%) when Text-CoT is used; this result replicates the finding in

McCoy et al. (2023) that CoT is helpful for shift ciphers but still remains far from perfect. However, with Number-CoT, GPT-4's performance becomes nearly perfect (more details are in Appendix A.1).

These results paint CoT prompting in a puzzling light. Prompting with Number-CoT showed that GPT-4 has the core reasoning abilities that would be needed to decode shift ciphers nearly perfectly. Thus, if CoT prompting led to symbolic reasoning, GPT-4 would score perfectly. The fact that it does not shows that CoT reasoning is not pure symbolic reasoning. Nonetheless, it is also clear that CoT does substantially improve over standard prompting, so it is unlikely that CoT reasoning can be explained away as simple memorization. If CoT reasoning is neither simple memorization nor pure symbolic reasoning, what is it? This question motivates our experiments in the next section.

4 Disentangling the factors influencing CoT performance

We consider four types of reasoning processes that LLMs might be adopting.

(a) **Symbolic reasoning** is the use of discrete, deterministic inference rules. Shift ciphers can be perfectly decoded with a simple symbolic algorithm, so a system using fully systematic reasoning should attain 100% accuracy.

(b) **Noisy reasoning** is like symbolic reasoning but with the addition of noise that introduces some possibility of each intermediate operation in a reasoning step being wrong. Thus, if the system uses noisy reasoning, we should see accuracy decrease as we increase the number of operations that need to be performed. Shift ciphers let us test this possibility: by varying $shift\_level$, we can modulate the number of operations that need to be performed in every reasoning step and observe if accuracy varies accordingly.

(c) **Memorization** is a strategy in which a system memorizes the tasks it has encountered in pre-training but does not generalize to new tasks. If memorization is all that LLMs do, we should see higher performance in the cases that are frequently encountered during pre-training than the ones that are not. McCoy et al. (2023) show that 13 is by far the most common shift level in natural corpora because this shift level (sometimes called `rot-13`) is popular in some online communities. Thus, a hallmark of memorization would be much higher accuracy at 13 than other shift levels.

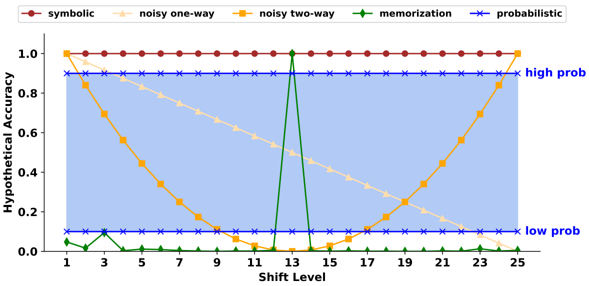


Figure 4: **Hypothetical** accuracy vs. shift-level for various types of reasoning. Under noisy one-way, the model only shifts letters backward; under noisy two-way, it adopts the shorter path between going forward and backward. The hypothetical memorization accuracy is based on shift level frequencies in internet corpora. Probabilistic would involve much higher scores on high prob than low prob.

(d) **Probabilistic reasoning** frames a task as choosing the output that is most probable given the input. Such reasoning would be influenced by the prior probability of the output: a probabilistic reasoner should show accuracy that increases as the prior probability of the correct answer increases.

Figure 4 illustrates the hypothetical performance trends that would be observed in a system adopting each reasoning approach. These approaches are not mutually exclusive; e.g., a reasoner could be influenced by both probability and memorization. Indeed, as discussed below, we find that LLMs' performance when prompted with CoT displays hallmarks of several different types of reasoning; see Figure 1, panel 4 for GPT-4 and Figure 11 in the Appendix for Claude 3 and Llama 3.1.

First, accuracy generally decreases as the shift level increases, a hallmark of noisy reasoning. LLMs' performance is, in particular, indicative of a two-way version of noisy reasoning in which it can decode a message by shifting letters forward or backward (e.g., instead of decoding a shift of 25 by shifting back 25 letters, it could instead shift forward 1 letter, as doing so requires fewer steps); this two-way nature shows up in the way that accuracy increases as the shift level changes from 20 to 25.

Second, evidence of probabilistic reasoning can be seen in the fact that accuracy is substantially higher for high prob (the highest-probability bin, `bin 1`) than low prob (the lowest-probability bin, `bin 5`). High prob / low prob refer to the probability of the words that are the correct answers when our examples are decoded, where probability is quantified using GPT-2 as described in Section 3.2. The “high prob” cases are common words such as *{’mariner’, ’shrines’, ’paywall’, . . . }*, while the “low prob” cases are nonsense letter sequences such as *{’xcbrouw’, ’jsxrouw’, ’levjspx’, . . . }*.

Finally, although a shift level of 13 requires the most reasoning steps of any shift level (assuming decoding can be done forward or backward), there is a spike in accuracy at a shift level of 13. As discussed above, this spike is a hallmark of memorization since 13 is the most common shift level in natural corpora.

For the upcoming detailed analysis experiments, we use GPT-4 as the main reference model, since Claude 3 and Llama 3.1 exhibited similar trends as GPT-4 on the evaluations presented so far.[*]

4.1 A simple probabilistic approach to modeling the reasoning process

To make these intuitively-stated observations more rigorous, we perform a logistic regression to determine the statistical significance of several factors. The outcome variable is a binary variable indicating whether GPT-4 got the correct answer on each example. We include the following predictors:

- $\mathbf{input\_logprob}$: log probability of the encoded input text as measured by GPT-2 (Radford et al., 2019). The inputs tend to have a very low probability because they are enciphered.
- $\mathbf{output\_logprob}$: log probability of the ground-truth output text as measured by GPT-2.
- $\mathbf{shift\_freq}$: we used the frequency of occurrence of all shift levels that McCoy et al. (2023) provided based on analysis of the C4 corpus (Raffel et al., 2020). The assumption is that the distribution of shifts in C4 is similar to the distribution in the training data for GPT-4.
- $\mathbf{shift\_level}$: the number of steps that must be performed to decode each letter; this feature is added to account for one-way reasoning.
- $\mathbf{min\,(shift\_level, 26} - \mathbf{shift\_level)}$: this value is the minimum number of steps that must be performed to decode each letter, under the assumption that decoding can be done by moving $\mathit{shift\_level}$ steps backward or $(26 - \mathit{shift\_level})$ steps forward; as discussed above, GPT-4 indeed shows evidence of using both of these decoding directions.

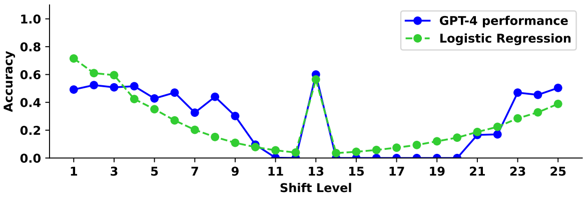


Figure 5: The logistic regression curve captures the overall trend exhibited by GPT-4.

Several of these variables correspond to the critical properties that are indicative of our hypothesized reasoning processes: $\mathit{output\_logprob}$ should have a significant effect if probabilistic reasoning is used, $shift\_freq$ should have a significant effect if memorization is used, and $\min(shift\_level, 26 - shift\_level)$ quantifies the difficulty of the task, which should have a significant effect if noisy reasoning is used. The remaining factors are included as potential confounds to control for. The overall logistic regression thus took the following form:

$$
\begin{aligned}
\mathit{correct} \sim\ & \mathit{input\_logprob} + \mathit{output\_logprob} \\
& + shift\_freq + shift\_level \\
& + \min(shift\_level, 26 - shift\_level)
\end{aligned}
$$

Logistic regression results. The following features had a statistically significant effect on model performance: $\mathit{output\_logprob}$, $shift\_freq$, $shift\_level$, and $\min(shift\_level, 26 - shift\_level)$ ($p < 10^{-15}$ in all cases). These results therefore quantitatively support the conclusion that GPT-4 incorporates processes based on probability, memorization, and noisy reasoning (both forward and backward).

Figure 5 shows the predictions of the logistic regression compared to GPT-4’s actual performance. The logistic regression correctly predicts the main trends in GPT-4’s behavior (as expected, it does not match the curve exactly due to the simplicity of the model). In the next few subsections, we conduct some additional experiments that investigate each hypothesized reasoning type in more detail.

4.2 Analyzing the effect of probability

If an LLM is influenced by probability, we would expect to occasionally observe *unfaithfulness* between the chain of reasoning steps produced by the LLM and the LLM’s final answer. Specifically, if the individual reasoning steps would point to a final output that is low-probability, a probabilistic reasoner might instead produce a different final answer that has a higher probability. For example, in

[*] The detailed results of `Llama-3.1-405B-Instruct` and `claude-3-opus-20240229` are shown in Appendix §A.2.

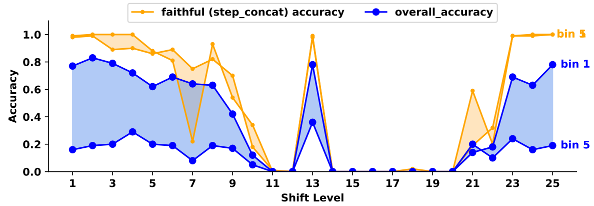


Figure 6: **Actual** overall decoding accuracy vs. *faithful* accuracy across shift levels showing the effects of probability. The effect is amplified for low probability outputs as seen in the larger drop in accuracy between the orange and blue `bin 5` (low probability) lines.

our CoT experiments, each step produces one letter, and these letters must be concatenated to form the final answer. If the individual step outputs are *S, T, A, Z*, the final answer should be *STAZ*, but a model might instead "self-correct" by producing the higher-probability word *STAY*.

Such unfaithfulness can help or hurt the model. When the correct answer truly is a low-probability word such as *STAZ*, then correcting to *STAY* would reduce the model's accuracy. However, if the model had made a mistake during the reasoning chain—such as by producing *S, T, A, Z* when the chain should have been *S, T, A, Y*—then correcting to *STAY* would rescue the model from its error.

To investigate unfaithfulness, we compare the *faithful accuracy* that would be obtained by concatenating GPT-4's step outputs to the actual overall accuracy. We indeed observe that overall accuracy is generally lower than faithful accuracy, illustrating that unfaithfulness occurs. Further, the drop in accuracy is more pronounced in the low-probability setting than the high-probability setting, which is consistent with the intuition that the lower the probability of a concatenated answer is, the more likely it will be that a probability-reliant model will be steered away from that answer. See Figure 6 for the full results.

Table 1 provides a more detailed view of unfaithfulness. Incorrect intermediate chains (i.e., concatenated step outputs) are followed by correct final answers much more often in the setting where the correct answer has a high probability (34% and 55% of the time for rot-4 and rot-13, respectively) than in the low probability setting (1% and 19% of the time for rot-4 and rot-13, respectively). On the other hand, correct intermediate chains are followed by incorrect final answers *less* often in the high probability setting (7% and 1% of the time for rot-4 and rot-13, respectively) than in the low-probability setting (14% and 9% of the time for rot-4 and rot-13, respectively).

Prob	Chain Steps Output	`rot-4`		`rot-13`	
		Correct	Incorrect	Correct	Incorrect
High	Correct	19	7	15	1
	Incorrect	34	40	55	29
Low	Correct	7	14	7	9
	Incorrect	1	78	19	65

Table 1: Confusion matrices (100 examples; 2 probability bins {high, low}) for `rot-4` and `rot-13`. *Effect of memorization*: incorrect step outputs lead to correct final answers more often for `rot-13` than `rot-4`. *Effect of probability*: Unfaithfulness has a positive effect more often in high-probability bins than in low-probability bins; Conversely, unfaithfulness has a negative effect more often in low-probability bins than in high-probability bins.

These results support the hypothesis that GPT-4 over-relies on the prior probability of potential outputs (see Jang et al. (2023) for some related observations). If the answer has a high probability of occurrence, the model's priors favor generating it even if its intermediate reasoning steps suggest an alternative output. Conversely, if the answer is of lower probability, then even if the chain of reasoning is correct, the priors exert a detrimental influence leading to incorrect final answers.

4.3 Analyzing the effects of noise

The statistically significant impact of $shift\_level$ is evidence that GPT-4's CoT behavior is in part a noisy version of symbolic reasoning. Accuracy falls as the shift level increases from 1 to 12 and then recovers at higher shift levels (Figure 8), consistent with a noisy reasoning process in which deciphering each letter with a shift level of n involves $\min(n, 26-n)$ implicit steps, with noise that gives each step some probability of being performed incorrectly. Note that the implicit steps referred to here are different from the steps that are explicitly produced in the chain of thought: the chain of thought uses one explicit step per letter (Figure 3), but here we are discussing the operations that must be implicitly carried out within each step of this chain in order to decode each letter.

Noise relating to complementary shift levels. We have argued that the relation between accuracy and shift level is evidence that GPT-4 uses a two-way strategy. That is, accuracy is high for small shifts such as 1 but also for large shifts such as 25, which could plausibly be explained by GPT-4 im-

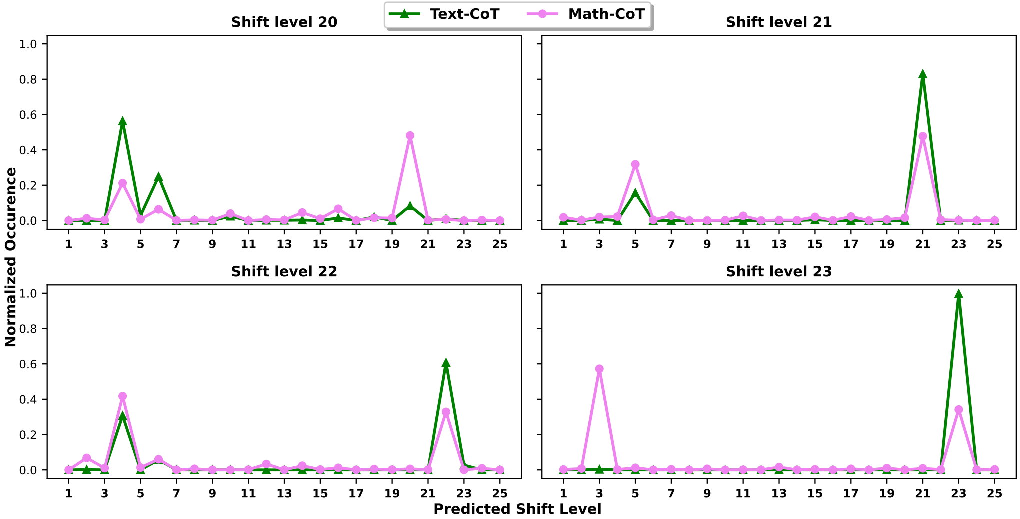


Figure 7: Normalized frequency distribution vs. predicted $shift\_level$ of step answers for `rot-20` to `rot-23`. The appearance of peaks at $26 - shift\_level$ in Math-CoT and Text-CoT prompts showcases the model's noisy attempt in taking the shorter path—i.e., moving $26 - x$ shifts forward.

plicitly selecting whichever direction will minimize the number of steps it needs to compute—shifting letters backward for small shift levels or forward for large shift levels. This two-way strategy is effective in that it supports strong performance on large shift levels such as 25. However, we also observe evidence that it contributes to the noise that causes accuracy to decline as the shift level increases. Figure 7 shows the actual shift level that GPT-4 produces for each letter in its chain of thought, for each of four intended shift levels. Across all four of these cases, GPT-4 shows peaks at both $shift\_level$ and $26 - shift\_level$. Thus, while some of the noise affecting the reasoning process may be random, it appears that at least some of the noise can be attributed to confusion between possible shift levels. Decoding a shift level of n can be done by shifting backward n steps or forward $26 - n$ steps, but it appears that GPT-4 sometimes mixes up these two strategies by shifting forward n steps (or, equivalently, shifting backward $26 - n$ steps), contributing to the overall noise.

Discretization. Another factor that interacts with noise is `temperature`. In principle, if CoT entailed pure symbolic reasoning, it would assign 100% probability to the correct continuation (i.e., each predicted next token) and 0% probability to everything else. If so, the temperature used would not affect performance. However, we observe that CoT scores better with a low temperature. For instance, in `rot-13`, GPT-4's accuracy is 30.0% at `temperature=0` and 0.33% at `temperature=1`.

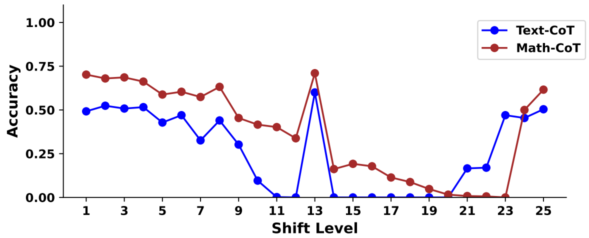


Figure 8: Accuracy for Text- and Math- CoT prompt styles with GPT-4. Math-CoT performs better than Text-CoT, but both display evidence of memorization as accuracy is highest for shift level 13—the most frequent shift in real-world corpora.

This shows that its predicted distribution over the vocabulary is not fully discrete—it has some noise in it that a low temperature can remove. However, even a temperature of 0 does not make the performance perfect because noise does not solely arise in the final distribution over the vocabulary (which temperature modifies) but also influences the implicit intermediate steps used to produce that distribution (which temperature does not change).

4.4 Analyzing the effect of memorization

To further investigate memorization, we focus on `rot-13`, because frequency is generally confounded with simplicity for the other shift levels (e.g., `rot-1` is simple as well as frequent). 13 is the most frequent shift level (McCoy et al., 2023), and we observe in Figure 8 that GPT-4 shows a spike in accuracy at this shift level in both Text-CoT and Math-CoT, providing strong evidence that memorization plays a role in GPT-4's CoT performance.

We also observe that memorization influences *unfaithfulness*. Consider the cells in Table 1 that involve incorrect chain steps outputs but correct final answers; such cases are much more common for `rot-13` than for other levels, including `rot-4` (the other shift level shown in that table); e.g., in the high-probability case, 55% of rot-13 examples fall in this category, while only 34% of rot-4 examples do. This pattern also provides some evidence for memorization: for `rot-13`, the model may have two "paths" for producing the final output—it could use the chain of thought it has produced, or it could go directly from the input to the output due to memorization. Thus, when it produces the final output, it might implicitly weigh both of those paths, which helps it to correct faulty chains because it has a back-up path to consider. However, for `rot-4`, it may be that only the path involving the chain of thought is available, such that GPT-4 cannot fix incorrect chains as easily because it does not have this alternative path to fall back on.

4.5 The role of intermediate reasoning steps

Finally, we study the role of the intermediate reasoning steps that are involved with CoT prompting—both the chain that GPT-4 produces and the chain provided in the demonstration.

Strong reliance on the surface strings produced in reasoning steps. First we focus on the chain of thought that GPT-4 produces before providing its final answer. We consider two potential roles that this chain could have. First, it could be that the chain is helpful because it provides text that is useful for GPT-4 to condition on in later steps. Alternatively, it could be that the critical aspect of CoT reasoning is internal—rather than depending on the text that is produced, CoT could be helpful because it gives the LLM the opportunity to internally perform additional reasoning steps.

To disentangle these possibilities, we modify the prompt so that GPT-4 is told to perform the same steps of reasoning as before, but to have the intermediate output that it produces be uninformative. Specifically, we used the Text-CoT prompt but instructed the model to *not reveal step answers and instead output a* *. The step answers in the demonstration were also replaced by '*'; thus we left the format of reasoning intact but the expected generation token was no longer a component of the final answer. Next, we asked the model to explicitly *think about the correct letter that should go in the place of the * but just not write it down*. In the demonstration, we first provided an example with all step answers and then repeated the same example but with a * in place of each output letter (see Appendix A.3 Figure 12 for prompts).

In both settings, performance was similar to that of the standard prompting variant (shown in Figure 1, panel 2). This is evidence that CoT depends on "self-conditioning"—explicitly producing text that will be useful as context to condition on when producing the final answer. Merely instructing a model to "think" silently is not helpful, suggesting that the reasoning does not occur internally. These results corroborate prior work finding that CoT is unhelpful when the model is told to produce contentless "filler" tokens instead of contentful text (Lanham et al., 2023); models can reason internally when explicitly trained to do so (Pfau et al., 2024), but current LLMs without this explicit training do not seem to have this ability.

Little reliance on the validity of the demonstration. In experiments described in Appendix A.3, we also find that the validity of the reasoning shown in the prompt does not have a strong effect of CoT performance for shift ciphers. That is, even when the demonstration is perturbed such that it contains many errors, GPT-4's CoT performance remains approximately the same. This finding corroborates prior work showing that the validity of demonstrations did not matter much (Wang et al., 2023a; Madaan and Yazdanbakhsh, 2022; Ye et al., 2023); the demonstration seems to merely guide the model to solve the task by providing a format to generate accurate reasoning steps (Min et al., 2022).

5 Conclusion

We have used the case study of shift ciphers to disentangle the factors that influence CoT reasoning, with a focus on characterizing what *type* of reasoning is used in models prompted with CoT. We found that CoT performance is statistically significantly influenced by the probability of occurrence of the expected task output, the frequency of the task in corpora, and the number of reasoning steps that must be (implicitly) performed. These results suggest that CoT reasoning can be characterized as probabilistic, memorization-influenced noisy reasoning, meaning that LLM behavior displays traits of both memorization and generalization.

Limitations

We have conducted extensive studies on a single task, decoding shift ciphers; we chose this task because it enables us to separate memorization from reasoning in a controlled manner as explained in Subsection 3.1, and these factors cannot be easily disentangled for most other tasks. Our prompts contain one example demonstration (i.e. one-shot CoT prompting), but a single demonstration contains multiple reasoning steps which provide more than just one reference of decoding to the model. In addition, while our experiments showed that these frontier models display some hallmarks of true reasoning, they also showed some ways in which they make errors due to a reliance on memorization and probability; future work could investigate how these limitations could be overcome.

Ethical Considerations

We do not believe that this work raises major potential risks. As is the case with any analysis work, there is some risk that our results could lead some readers to overestimate or underestimate the abilities of LLMs, either of which could have negative consequences: overestimation can contribute to hype, while underestimation can result in the field paying too little attention to potential harms. However, we believe that this risk is minimal because we have aimed to present our results in a balanced way that highlights both strengths and limitations.

Acknowledgements

This work was supported in part by the National Science Foundation SBE Postdoctoral Research Fellowship under Grant No. 2204152. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Jason Andress. 2014. Chapter 5 - cryptography. In Jason Andress, editor, *The Basics of Information Security (Second Edition)*, second edition, pages 69–88. Syngress, Boston.

Anthropic. 2024. Claude 3. `https://www.anthropic.com/news/claude-3-family`. Accessed: 2024-08-03.

Antonis Antoniades, Xinyi Wang, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2024. Generalization vs. memorization: Tracing language models' capabilities back to pretraining data. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Boxi Cao, Qiaoyu Tang, Hongyu Lin, Shanshan Jiang, Bin Dong, Xianpei Han, Jiawei Chen, Tianshu Wang, and Le Sun. 2024. Retentive or forgetful? Diving into the knowledge memorizing mechanism of language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14016–14036, Torino, Italia. ELRA and ICCL.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 18878–18891. Curran Associates, Inc.

Ting-Rui Chiang and Dani Yogatama. 2024. Understanding in-context learning with a pelican soup framework. *arXiv preprint arXiv:2402.10424*.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. 2004. *Reasoning about Knowledge*. MIT press.

Vitaly Feldman. 2020. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Advances in Neural Information Processing Systems*, volume 36, pages 70757–70798. Curran Associates, Inc.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R.

Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2023. Can large language models truly understand prompts? A case study with negated prompts. In *Transfer Learning for Natural Language Processing Workshop*, pages 52–62. PMLR.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1830–1842, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Jannik Kossen, Yarin Gal, and Tom Rainforth. 2024. In-context learning learns label relationships but is not conventional learning. In *The Twelfth International Conference on Learning Representations*.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Hannah R. Sheahan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. 2024. Language models, like humans, show content effects on reasoning tasks. *PNAS Nexus*, 3(7):pgae233.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Yingcong Li, Kartik Sreenivasan, Angeliki Giannou, Dimitris Papailiopoulos, and Samet Oymak. 2023. Dissecting chain-of-thought: Compositionality through in-context filtering and learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew Hardy, and Thomas L. Griffiths. 2023. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*.

William Merrill and Ashish Sabharwal. 2024. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*.

MetaAI. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. Association for Computational Linguistics.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Yasaman Razeghi, Robert L. Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of EMNLP*, pages 840–854.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Saurabh Srivastava, Annarose M B, Anto P V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. 2024. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv preprint arXiv:2402.19450*.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? An analysis of CoT in planning. *arXiv preprint arXiv:2405.04776*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Peter C. Wason. 1968. Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20(3):273–281.

Peter Cathcart Wason and Philip Nicholas Johnson-Laird. 1972. *Psychology of reasoning: Structure and content*, volume 86. Harvard University Press.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Skyler Wu, Eric Meng Shen, Charumathi Badrinath, Jiaqi Ma, and Himabindu Lakkaraju. 2023. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions. *arXiv preprint arXiv:2307.13339*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Advances in Neural Information Processing Systems*.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4469–4484, Toronto, Canada. Association for Computational Linguistics.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023a. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. 2023b. What and how does in-context learning learn? Bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

```
Math-CoT

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
...

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

Then map the found position to the corresponding letter using the
letter-position mapping.

Using this,
1. f -> (5 - 13) mod 26 = 18 -> s
2. g -> (6 - 13) mod 26 = 19 -> t
3. n -> (13 - 13) mod 26 = 0 -> a
4. l -> (11 - 13) mod 26 = 24 -> 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 9: Math-CoT prompt.

```
Number-CoT

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: "5,6,13,11"

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: "18,19,0,24"

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: <test_input>
```

Figure 10: The shift cipher task reformatted to be performed in the domain of numbers rather than the domain of letters (Number-CoT). Here both the encoded input and decoded output are number sequences abstracting out the alphabet and its influences.

A Additional Details & Experiments

A.1 Math- & Number- CoT prompts

While the Standard prompt (Figure 2) yields poor performance on most shift levels, the Number-CoT prompt (Figure 10) in contrast gives nearly perfect scores across the shift levels. It is to be noted that some miscellaneous noise is captured in the Number-CoT case. This arises mostly due to incomplete generations/half-completed chains requiring more tokens than needed as the model does some additional sub-reasoning steps, and in very rare cases produces numbers greater than 25. The Math-CoT prompt used is shown in Figure 9.

A.2 Results with Llama 3.1 and Claude 3

We display the results with Llama 3.1 and Claude 3 when prompted with standard prompts and Text-CoT in Figure 11. Interestingly, Llama 3.1 was trained on CoT data, such that even when it is prompted with standard prompts it generates reasoning steps. However, as we can see from Figure 11, with Text-CoT there is a further enhancement in scores across the shift levels. Overall, both models display trends similar to that of GPT-4 shown in Figure 1.

A.3 Impact of producing explicit reasoning steps and validity of steps in demonstration.

Figure 12 shows the prompts used to test the importance of producing explicit reasoning steps by forcing the model to “think” silently.

Outputs from *silent thinking*. We observe that many of the outputs produced in the case when the reasoning step answers are * are related to the task of shift ciphers. It seems that the model now relies on terms such as *cipher* and *decode* that are present in the description, perhaps relying less on the reasoning steps because these reasoning steps are now less informative given that each step’s output is now *. Influenced by the probabilistic effects discussed before it produces outputs having a high probability and containing words related to security, safety, and programming such as *encryption*, *code cracker*, *decoded*, *Javascript*, and *Instagram*.

Validity. Next we consider the chain that is provided to GPT-4 within the prompt. Prior work has found that the validity of the reasoning shown in the prompt does not have a strong effect on CoT performance; CoT is approximately as helpful when the chain of thought in the prompt contains errors as when it does not (Wang et al., 2023a; Madaan and Yazdanbakhsh, 2022; Ye et al., 2023). Here we test whether this observation extends to shift ciphers.

We first tried applying random corruptions to the steps in the demonstration. Each step answer in the demonstration was replaced by a random letter; the format of the chain was left unchanged, with only the produced letters being modified. We also tried applying systematic corruptions by having the linguistic description of the task specify `rot-13` yet having the demonstration illustrate `rot-14`. Figure 13 shows the prompts used.

With both corruptions, GPT-4’s performance re-

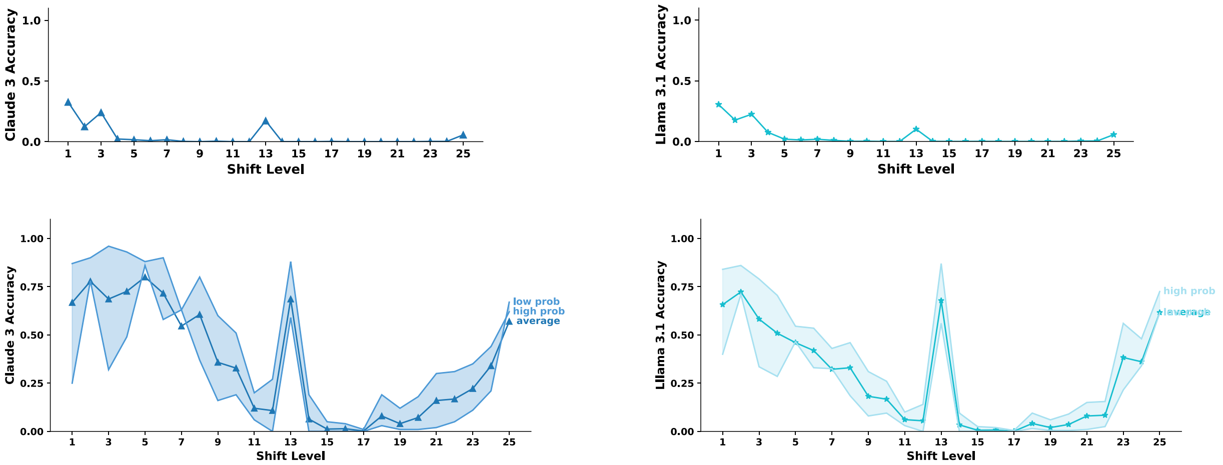


Figure 11: Accuracies with Claude 3 and Llama 3.1. The top panel shows the performance when prompted with Standard prompting. The bottom panel shows the trend with Text-CoT. The shaded regions indicate the gap between the low and high probability bins.

Hidden 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; but instead of revealing what each letter becomes, we will replace it with a '*'
until we write the final answer:
1. f -> *
2. g -> *
3. n -> *
4. l -> *
If we put together the letters that were hidden behind each '*', we get that 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>
```

Partially hidden 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

But, in order to preserve secrecy, we will ask you to write your answer in a slightly different way: Instead of revealing what each letter becomes,
we will replace it with a '*' until we write the final answer. Using this format, the reasoning shown above becomes:
1. f -> *
2. g -> *
3. n -> *
4. l -> *

Therefore, the original text is: "stay"

Here is another message in rot-13. Decode this message one letter at a time. As you produce each '*', we want you to still think about the correct
letter that should go in the place of the star - just don't write it down. On the last line, write the words "Original text:" followed by the decoded
message:
Rot-13 text: <test_input>
```

Figure 12: Text-CoT prompt consisting of hidden (top) and partially hidden (bottom) reasoning steps.

```
1. Text-CoT: rot-13 description with rot-14 demonstration

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: "ghom"

To decode this message, we shift each letter 13 positions backward:
1. g -> s
2. h -> t
3. o -> a
4. m -> y
Therefore, the original text is: "stay"
```

```
2. Math-CoT: rot-13 description with rot-14 demonstration

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: "ghom"

Using this,
1. g -> (6 - 14) mod 26 = 18 -> s
2. h -> (7 - 14) mod 26 = 19 -> t
3. o -> (14 - 14) mod 26 = 0 -> a
4. m -> (12 - 14) mod 26 = 24 -> y
Therefore, the original text is: "stay"
```

```
3. Random corruptions in demonstration

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 -> w
2. g -> a
3. n -> b
4. l -> i
Therefore, the original text is: "stay"
```

Figure 13: Prompt snippets with different types of mismatches/corruptions. The corruptions that were introduced are highlighted in yellow.

mained approximately as strong as when it was given accurate, uncorrupted demonstrations. This result corroborates the aforementioned prior work that found no strong correlation between the validity of the reasoning shown in the demonstration and the model performance. The demonstration seems to merely guide the model to solve the task by providing a format to generate accurate reasoning steps (Min et al., 2022).