What makes a benchmark effective for evaluating self-correction-based reasoning frameworks?

Anonymous ACL submission

Abstract

Recently, large language models (LLMs) have demonstrated remarkable reasoning abilities, augmented by recent advances in prompting techniques and reasoning frameworks. Many popular frameworks (Du et al., 2023; Yao et al., 2023; Shinn et al., 2023) rely on the assumption that models are able to give effective feedback on their own generations. This feedback is partly predicated on being able to correctly validate, or classify, the generated prediction as either correctly or incorrectly solving the given problem. While in traditional computer science settings validation has been shown to be as difficult as correct generation, we find empirically that language models may be better validators than generators. Our work studies whether leading language models are better at solving problems or validating solutions, and we attempt to gain a better understanding of why this happens. We quantify this by measuring the gen-val gap — the difference between generation and validation accuracy. First, we further corroborate recent work (West et al., 2024) showing surprisingly that models are better generators than validators on some datasets. Second, we discover that the generator-validator gap can vary significantly between skills (e.g. arithmetic, multihop reasoning, commonsense, etc.). Third, we apply our findings to suggest a combination of skills that should be represented in a dataset used for evaluating reasoning agent performance. This continues the conversation started by Huang et al. (2023), where we instead show that LLMs can self-correct reasoning, and establish a link between a feature of the dataset and the language model's ability to self-correct.[1]

1 Introduction

Language models have recently begun to demonstrate human-like reasoning capabilities, driven in large part by zero shot prompting-based self-correction algorithms. This brings to bear the question, how can we design consistent, and challenging, to assess model reasoning skills across so many different self-correction-based reasoning frameworks? And perhaps more fundamentally, why does the research community expect — and indeed observe — that language models can improve upon their initial generation? In this paper, we seek to begin an exploration into these two questions that we hope will support broader research interest.

To the former, many reasoning frameworks have begun to outgrow the previously commonly-used frameworks due to increasingly strong performance. This has led to many recent self-correction, multiagent debate, and self-consistency papers to evaluate on custom tasks (Yao et al., 2023; Shinn et al., 2023; Du et al., 2023).

On the latter, some researchers view self-correction as a form of post-hoc prompting (Huang et al., 2023), where generation quality improves through self-conversation quite simply because, within the confines of its context, the model is generating a better "prompt" for itself. This correlation between better prompting methods and higher generation accuracy has been demonstrated in the literature (Wei et al., 2022c,a), further supporting the strengths of this explanation.

In this view, self-correction would be viewed as several, iterative turns of a model generating prompts for itself. If this is indeed true, the recent success of self-correction methods would mean that our current prompting methods still fall short of optimal prompts. This would highlight the relevance of pursuing additional research into prompting and interpretability of language model self-corrections. While self-correction methods work well in-practice, we may not fully understanding why the model chooses to issue the corrections it does, and how it chooses which to accept.

A question of efficiency also arises. Many self-

[1] Code available at https://github.com/gen-val/gen-val

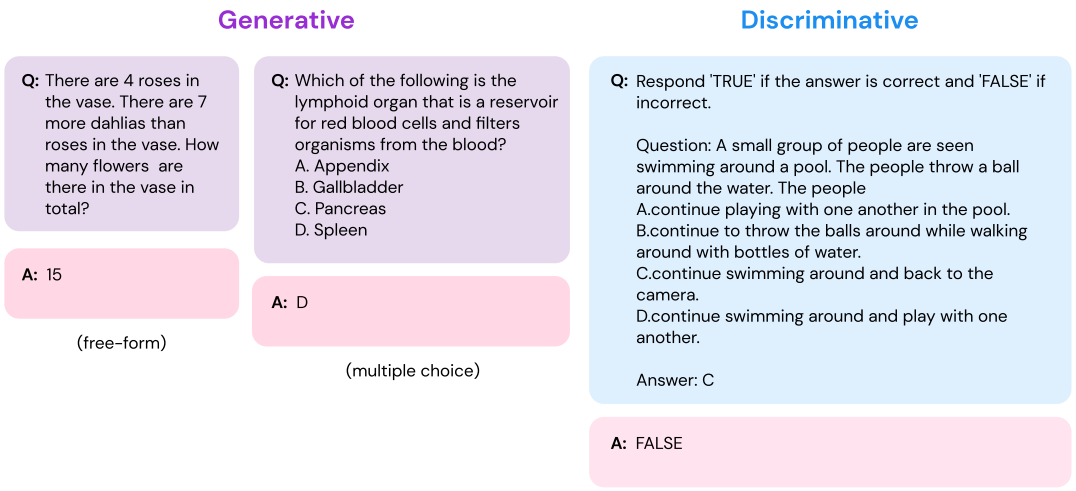

Figure 1: How do we assess "generation" and "validation" abilities of models? For the generative modality, we provide the model with free-form and multiple choice problems and assess based on solution accuracy. For validation, we provide the model with a question-answer pair and ask the model to assign a binary "TRUE" or "FALSE" label depending on whether it believes the predicted answer p to be correct given the question q.

correction algorithms are more computationally expensive than generating the correct answer directly from a human-specified prompt. Can we simply find the optimal prompt and forgo this expensive self-correction entirely? Ostensibly, by construction the use of this so-called optimal prompt would mean that no level of self-correction would improve upon the quality of the generation; whether measured as accuracy, determinism, or safety.

It is possible that there is a ceiling on the degree to which self-correction can improve accuracy. Otherwise, if the optimal prompt could achieve 100% accuracy across all datasets, it would seem prudent to direct significant focus towards prompt design. Until we have both better understood prompting, and we have designed or identified the models that can achieve this level of capability, however, we will likely have to rely on continued self-prompting. These ideas seem to suggest that models are not learning new things from self-correction but simply positioning themselves in a way to better retrieve existing knowledge.

We identify two broader modalities of generative model output: generation and validation (see Figure 1). Generation involves the structure of task used by most datasets: the language model is provided with a free-form or multiple choice question and is asked to generate the correct solution. The model is then evaluated on the accuracy of its solutions. In the validation setting, the model is provided the question and a prediction, then is asked to answer TRUE or FALSE depending on whether the answer correctly follows from the question.

This paper seeks to better grasp how and what language models understand, and how we can leverage these learnings when designing benchmarks. Given that self-verification, reflection, and most multiagent debate algorithms rely on some degree of validation-based correction, we hope that our study on relative generator-validator performance will help better understand why and when feedback is effective. It may also shed light on whether the bottleneck for multiagent systems is not path exporation, but instead reasoning ability. We refer to this generator-validator gap the gen-val gap and seek to find an ϵ that can upper bound the gen-val gap across all datasets. We will determine a reasonable estimate for this bound by assessing results across 10 popular datasets covering a diverse range of skills spanning constraint satisfaction, arithmetic, multihop reasoning, common sense, and reading comprehension. While it is expected that manipulating prompts can improve performance, the question is whether we can completely close the gen-val gap through prompting. We further attempt to apply our findings to predict the settings where self-correction is most effective, further engaging the hypotheses presented by Huang et al. (2023) and West et al. (2024). We specifically test the performance of multiagent debate (Du et al., 2023), which relies on self-correction, and use our learnings to better understand the types of datasets that

the algorithm performs most effectively on.

2 Related works

Prior work shows that language model performance can depend substantively on the structure of the prompt used. This begs the question of what it means for language models to understand, and whether we can conflate memory with understanding. It is possible that models are simply so good at memorizing and interpolating between these memorized answers that they appear to give semblances of understanding. Surely we wouldn't say that a grade school student has fully grasped algebra if they simply score well on problems similar to ones covered in class. That would just be memorization. Instead, we expect them to perform well on never-before-seen questions probing understanding, before we can confidently state that the student understands algebra. Lastly, we look at influential prior work whose effectiveness may depend upon the assumption that language models are better validators than generators.

Impact of prompting on model performance Prompting can remarkably allow models to adapt to new scenarios even with no task specific data (Wei et al., 2022a). Extending this further, the prompts themselves can be generated by models, which has shown to improve generative performance as compared to human-generated prompts (Gao et al., 2021; Guo et al., 2022; Ben-David et al., 2022). The use of previous generations as a prompt for future generations presents a recurring motif underlying much of the self-reflection and agentic debate space. However, current prompting methods are largely based on classification and generation, highlighting the need for more research on prompting for information extraction, text analysis or other interrogative understanding based tasks like validation (Liu et al., 2023).

Do language models understand deeply? Prior work has found that some language models have begun to show reasoning capabilities resembling a human-like general intelligence (Bubeck et al., 2023). While some studies have shown some of these emergent behaviors to be artifacts of dataset quirks (Wei et al., 2022b), other studies have more carefully investigated how human and model understanding may differ despite comparable generative capabilities (West et al., 2024). Other work has focused on whether language models are able to leverage this understanding to self-correct their generations to improve accuracy or morality (Huang et al., 2023; Ganguli et al., 2023) and have used genval consistent generations as a finetuning dataset to improve generator-validator consistency (Li et al., 2024).

Methods relying on strong validator capabilities A number of effective methods reliant on strong model validation have emerged in the literature. The first of two types involves improving factuality of model generations by using self-generated verification questions or through multiagent debate which rely on the ability of models to probe their own or each other's understanding through generations with varying priors (Dhuliawala et al., 2023; Du et al., 2023). Inspired in part by the increase in model reasoning ability when using its own generations as scaffolding for future generation (Wei et al., 2022c), the second of these types involves improving reasoning through prompts that allow the model to self reflect (Shinn et al., 2023; Yao et al., 2023; Madaan et al., 2023).

3 Are language models better generators or validators?

We attempt to formally investigate a phenomenon that may often be taken as granted — that language models are better generators than validators. This belief may arise from our expectations that the training data used by these models more closely resembles the generation task paradigm than the validator one. Notably, we specifically seek to gain a generalized understanding beyond single datasets, realizing that while models may achieve very strong validation performance relative to generation on some datasets it is much more helpful to show that we can get comparable validator-generator performance across most datasets through carefully crafted prompts.

Suppose we have a question q and a prospective answer a. For instance, consider a question q from the GSM8K dataset, a prediction p, and the reference answer a (see Table 1). Note that the prediction a may be different from the ground truth, as it is in this case. The goal of the validator is to identify whether the prediction p is correct, given the question q. The ground truth validator response is `TRUE` if $p = q$ and `FALSE` otherwise.

Intuitively, we expect models to have validation abilities that surpass its generative ones. This is because given q and p, we can always ask the model

q	Jame will turn 27 in 5 years. In 8 years his cousin will be 5 years younger than twice his age. How many years separate the age of the two now?
p	33
a	25

Table 1: Example of a (q, p, a) tuple where q is the question, p is the prospective answer, and a is the ground truth answer. Note that p is actually an incorrect answer in this case.

to predict the correct answer to q, then compare this with p. For a simple TRUE/FALSE validator, this would give us a validation accuracy identical to the generation accuracy. Surprisingly, for a generator with accuracy below 50%, this style of prompting would give us an accuracy that is worse than random guessing between the two TRUE, FALSE options (50%).

However, if we want to limit models to roughly as many tokens when generating vs. validating, we would no longer be able to use this approach, since validation would necessarily take more tokens than generation. This might cause the model to generate more hastily, likely reducing generation quality, since it still needs token space to perform the comparison between its generated answer and the prediction p. This is why we might expect that, in the worst case, that validator performance might trail generator performance slightly. However, if a language model is much better at generating than validating, we might suspect that it has a bit of an understanding gap — perhaps the model is generating answers without fully understanding why it is correct. We can create a metric attempting to capture the model's "understanding" ability. Formally, we define this term `gen-val gap` to be generation accuracy less validation accuracy of a language model on a specific task.

3.1 Stochastic parroting with interpolation

Large language models model closely their training data, a phenomenon coined as stochastic parroting (Bender et al., 2021). It is thus contendable that on some types of problems, the language model may simply be parroting the answers that lie in its training data, interpolating between multiple datapoints when the exact phrasing of the question differs. Since this suggests that the language model is relying more on pattern recognition rather than true understanding to solve the problem, we would suggest poorer understanding and thus a higher gen-val gap. If we can bridge this gen-val gap, we are likely also improving the model's reasoning ability for the problem and thus improve the usefulness of self-corrections.

Where the Generative AI paradox (West et al., 2024) provides a hypothesis based on relative validator performance between humans and models, we provide a hypothesis that investigates the validation ability of models relative to their own generation ability. The hypothesis guiding this paper is as follows:

Hypothesis 1: Models are, approximately, as good at validating solutions as they are at generating answers to questions. However, this performance requires the careful choice of an appropriate validator prompt. We state the hypothesis formally as,

$$\forall t \in T, \exists p_t \in P \texttt{ s.t. } g(t) - v(t, p_t) < \epsilon$$

where t is some task in the set of all possible tasks T, p_t is some task-specific prompt in the set of all possible prompts P, $g(t)$ is some function that captures the generative performance on a task t, $v(t, p)$ is some function that captures the validation performance on a task t with prompt p, and ϵ is some small non-negative value, the upper bound on the generator-validator performance spread across all tasks. We consider $g(t) - v(t, p_t)$ to be a formal definition of the gen-val gap. The quality of the function for $g(t)$ and $v(t)$ dictates whether the result of this hypothesis is trivial or useful. Specifically, we know that if $g(t)$ and $v(t)$ are naively chosen as % accuracy measures, then the hypothesis is true under a model that randomly guesses on free-response questions, since expected generation accuracy would be near 0% while validation accuracy would be close to 50%, since the model just needs to answer TRUE or FALSE during validation.

To make a strong case supporting this hypothesis, we desire to show that across a diverse range of datasets there exists an upper bound on generator-validator accuracy spread. To increase the robustness of our study, we also investigate two subhypotheses whose validity we expect to be consistent with Hypothesis 1. If these two sub-hypotheses are found to be concurrently valid with those of Hypothesis 1, we believe this presents a strong case for the latter. These two subhypotheses investigate

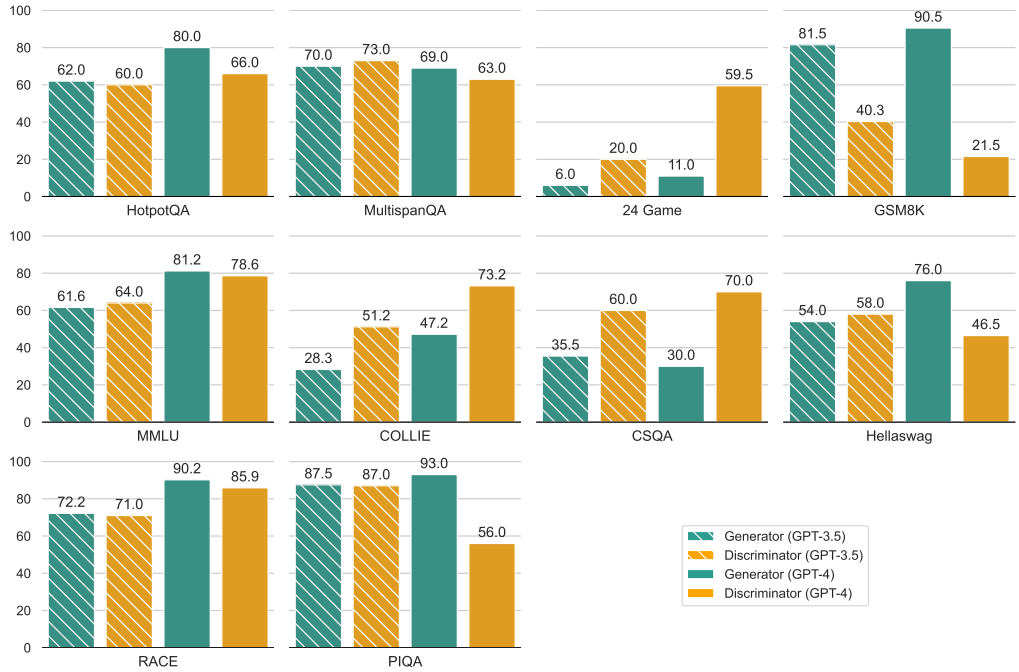

Figure 2: Generation and validation accuracy (↑) of GPT 3.5 and 4 across various skills, in the **standard** configuration

the ability of prompting to improve validation performance and whether validation accuracy captures underlying model understanding well.

3.2 Skill dependency of generative-validative capabilities

While it may seem trivial that skill or dataset type has an effect on generation quality, the extent and nature of the effect is less clear. Many popular prompting methods are designed for generation rather than validation settings, and it is unclear whether prompting improves generations simply because more computation is performed or whether it is unlocking an otherwise elusive understanding of the problem. Further, while a better prompt may improve accuracy on a dataset, it is uncertain whether, in the eyes of a model, a generation with a better prompt will allow a model to become a teacher to itself, producing a single model student-teacher model configuration.

We take a positive stance on the ability of prompting to tighten the upper bound. Specifically, since $g(t)$ and $v(t, p)$ are both measures of accuracy between 0 and 1, the trivial upper bound ϵ is 1. We also realize that some tasks may naturally have negative gen-val gaps because correct answers may be especially difficult to generate for both humans and machines, despite being fairly easy to validate. These tasks might, for instance, involve satisfaction of a simply stated mathematical or textual constraint that has a large state space of potential answers with non-intuitive mapping of the answer space to constraint satisfaction.

Sub-hypothesis 1: We can find variances in generator-validator performance across skills.

3.3 Crafting a challenging dataset

We next investigate whether the fact that a dataset has a tighter generator-validator gap will mean that self-correction-based algorithms will struggle more to produce the more effective self-reflections which may lead to greater overall performance improvements. If it does, the generator-validator gap presents a simple and objective metric to assess the difficulty of a dataset for these types of reasoning frameworks. If correct validation constitutes understanding, we would expect the following sub-hypothesis to be true. We leave this second sub-hypothesis as an open question to the research community.

Sub-hypothesis 2: Datasets with tight generator-validator gaps will present a more significant challenge to elf-correction-, self-consistency-, and multiagent-debate-based models, which may observe smaller gains.

4 Methodology

Benchmarks. We run experiments across datasets spanning various natural language skills. We investigate the ability of models to perform constraint satisfaction through **COLLIE** (Yao et al., 2024), arithmetic through **24 Game** (Yao et al., 2023) and **GSM8K** (Cobbe et al., 2021), and multihop reasoning via **HotpotQA** (Yang et al., 2018), **MultispanQA** (Li et al., 2022) and **MMLU** (Hendrycks et al., 2021). Equally important are the abilities to operate in settings of commonsense through **CSQA** (Talmor et al., 2019), **Hellaswag**, (Zellers et al., 2019) and **PIQA** (Bisk et al., 2020), and reading comprehension via **RACE** (Lai et al., 2017). We evaluate on uniformly randomly selected examples (n=100) from each dataset based on recent work showing that for many benchmarks, evaluation on n=100 curated examples achieves estimation errors under 2%, for example, in the case of MMLU (Polo et al., 2024).

Models. Since we are testing the limits of current state-of-the-art language model generation and understanding capabilities, we choose to evaluate on the most popular (among both researchers and industry users) and capable language models. To that end we evaluate our hypothesis primarily on **GPT-3.5** and **GPT-4**, specifically `gpt-3.5-turbo-1106` and `gpt-4-1106-preview`, respectively (John Schulman et al., 2022; OpenAI et al., 2023).

Evaluation. We assess model performance in standard and log probs settings. In the **standard** setting, for generation we directly prompt the model with q (see Table 1), extract the correct answer, and calculate accuracy by comparing with the ground truth. The latter two are performed programmatically, but to account for errors, the authors manually audit each generation. For validation, we present the model with the original question and its own generation and ask it to respond with `TRUE` if it believes the answers to be correct and `FALSE` otherwise.

While benchmarks are designed to be evaluated under the standard setting by default, we also assess using the **log probs** setting, a method we design to avoid surface form competition (Holtzman et al., 2021) by normalizing log probabilities of the correct generation over all valid options. These custom metrics also make the values more comparable. We first convert each task into a multiple choice problem, using GPT-4 to generate options if they do not currently exist. Next, for the validation modality, we generate the top 20 most likely answers to the question, based on log probabilities. From these answers, we look for the (answer, log probability) pairs that match with the multiple choice options. We then apply a softmax-like function for the correct answer over all valid options (Equation 1). This is how we score generations, and for the overall score of a model (between 0% and 100%), we take the mean of this score across all problems.

$$\text{score}_{\text{gen},z} = \frac{e^{z}}{\sum_{o \in O_q} e^{z_o}} \tag{1}$$

For the validation modality, we pass the question q, including options, and the prediction p to the model, and ask it to answer `TRUE` if it believes p to be the correct answer to q and `FALSE` otherwise. We do this for $p \in O_q$ where O_q is the set of all options for question q. We then apply a function similar to Equation 1 to produce a score for each option normalized across the two valid responses ($\in$ {TRUE, FALSE}),

$$\text{score}_{\text{disc},o,z} = \frac{e^{z}}{\sum_{o \in \{\text{TRUE,FALSE}\}} e^{z_o}} \tag{2}$$

where z is the log probs of the correct answer (either TRUE or FALSE) and z_o is the log probs for the output o. The mean of the correct selection for each option (`TRUE` for the correct answer and `FALSE` for the other choices) across all options is the validator score.

For our baselines (see Figure 2), we use a simple system prompt with a minimal amount of information necessary to instruct the model to complete each task, running on N=200 examples. To evaluate the effect of prompt, we try various system prompts following at times architectures shown to be effective in past works. In evaluating the effect of gen-val gap on reflection algorithms, we implement two popular techniques for self-correcting reasoning: Reflexion (Shinn et al., 2023) and Multiagent Debate (Du et al., 2023).

5 Discussion

5.1 Are language models better generators or validators?

Of the 10 tasks spanning skills in constraint satisfaction, arithmetic, multihop reasoning, common

	HotpotQA	MultispanQA	24 Game	GSM8K	MMLU	COLLIE	CSQA	Hellaswag	RACE	PIQA
gpt-3.5-turbo-1106										
Generation	0.5510	0.5806	0.2651	0.4997	0.5064	0.4322	0.5576	0.5979	0.4529	0.5259
Validation	**0.7136**	**0.6662**	**0.3285**	**0.7721**	**0.5670**	**0.6188**	**0.6702**	**0.7769**	**0.5317**	**0.5243**
gpt-4-1106-preview										
Generation	0.7854	**0.8370**	0.2673	0.7105	0.7833	0.6168	0.8033	**0.8246**	0.6383	0.6775
Validation	**0.8277**	0.8226	**0.6866**	**0.7603**	**0.8464**	**0.7237**	**0.8153**	0.8166	**0.7016**	**0.7583**
gpt-3.5-turbo-1106 (rescaled)										
Generation	0.4013	**0.4408**	0.0201	0.3329	**0.3419**	**0.2429**	**0.4101**	0.4639	**0.2705**	**0.3679**
Validation	**0.4272**	0.3324	**0.3430**	**0.5442**	0.1340	0.2376	0.3404	**0.5538**	0.0634	0.0486
gpt-4-1106-preview (rescaled)										
Generation	**0.7139**	**0.7827**	0.0231	**0.6140**	**0.7111**	**0.4891**	**0.7377**	**0.7661**	**0.5177**	**0.5700**
Validation	0.6554	0.6452	**0.3732**	0.5206	0.6928	0.4474	0.6306	0.6332	0.4032	0.5166

Table 2: Generation and validation performance of GPT-3.5 and -4 (softmax scores (↑)) in the **log probs** configuration. For each model and dataset, the max of the validation and generation scores is bolded. Since the random guessing accuracy of generation is 25% and validation is 50%, we also show rescaled values to map from the random guessing-to-perfect-performance range to the 0-to-1 range.

sense, and reading comprehension, 3 have negative gen-val gaps where validation performance surpasses generation performance on both models (see Figure 2). Surprisingly, we find that while GPT-4 generally outperforms GPT-3.5 in generative capacities, it sometimes surprisingly underperforms on validation on those same tasks, perhaps suggesting semblances of overfitting in GPT-4 to more oft seen dataset paradigms. In fact, GPT-3.5 presents an gen-val gap on only 4 of the 10 datasets, while GPT-4 presents a gap on 7 of the 10.

For the tasks with positive gen-val gaps on GPT 3.5, however, the gap is often small. The exception is GSM8K where the model significantly underperforms in a validation setting, trailing even random guessing (50% for a `TRUE/FALSE` configuration) despite high generation accuracy, suggesting anticorrelation. This is promising, since a trivial (and problematic on principle) solution could be to prompt the model to answer the opposite of what it thinks. Ostensibly, this would then give us a (100%-40.3%=59.7%) accuracy, but wouldn't really be faithful to our underlying exploration of model understanding.

In the baseline setting, the model exhibits gen-val gaps on multihop reasoning, arithmetic, reading comprehension, and common sense skills. In the log probs setting (see Table 2), we notice that in our original unscaled measure of accuracy, the validator almost always outperforms the generator. However, by rescaling we produce a more fair function for $g()$ and $v()$, which allows us to observe that GPT-4 is a consistently better generator than validator, except on 24-game.

5.2 Can we make up for the gen-val gap through prompting?

On GPT-3.5, we close the gen-val gap on HotpotQA by using prompt P1 (see Appendix A). We find remarkably that despite the model still only outputting a single token (`TRUE` or `FALSE`), validation accuracy surpasses the generator accuracy; closing the gap and even resulting in a negative gen-val gap. We are able to reduce both gen-val gaps for RACE and PIQA below 0.5%, however, the model continues to validate poorly on GSM8K, meaning our observed understanding bound ϵ remains large at 0.412.

On GPT-4, we close the gen-val gap on 2 datasets (HotpotQA and MultispanQA) and reduce the gap on 3 others to below 5.5%. However, validation continues to lag on GSM8K and Hellaswag datasets. In both HotpotQA and Multispan settings, we find remarkably that the model significantly outperforms the baseline validator as a result of prompting without outputting additional tokens.

We show that we can make up a substantial portion of the gen-val gap through prompting and are able to reach an upper bound on the gen-val gap across all 10 datasets of 0.412. This performance in addition to the near-zero gen-val gap produced suggests that it is possible to prompt the model to transition from reasoning-based to retrieval-based validation.

5.3 When do self-correction algorithms work better?

Do algorithms based on models' self-correction perform better on tasks with smaller gen-val gaps? While we have left this sub-hypothesis as an open

question, we find in the literature that the tasks with greatest improvement from prompting (41% boost over CoT in 24 game (Yao et al., 2023)). More curiously, however, some reflection-based algorithms 31% over CoT in MultispanQA (Dhuliawala et al., 2023)).

5.4 Two types of validation

We hypothesize the existence of two types of bases of validation: logical reasoning and retrieval. The first, logical-reasoning-based validation, is largely non-reliant on existing knowledge and validates a solution purely based on whether the solution is valid. For humans, this is equivalent to validating a Sudoku solution by checking whether each row, column, and sub-grid is duplicate-free. The second, retrieval-based validation, is knowledge-dependent (hence the "retrieval" moniker) and involves solving the problem, then comparing the solution to the provided answer. For humans, this is equivalent to validating the Sudoku solution by solving the puzzle then comparing the two solutions (assuming only a single solution exists for the puzzle).

The astute reader will likely be contemplating some key characteristics that we would expect to see if this hypothesis were valid. We would expect models relying on retrieval-based validation to fail when presented with problems with multiple correct solutions (e.g. constraint satisfaction tasks like COLLIE (Yao et al., 2024), and some Sudoku and 24 Game puzzles). A model using retrieval-based validation would be expected to perform approximately equivalently on generation and validation, since validation is simply generation plus a comparison. However, a model using logical-reasoning-based validation would be expected typically to more significantly under- or over-perform on validation as opposed to generation, since its validation ability is decoupled from its generation ability. We believe that when researchers discuss validation (or what we consider to be its synonym, *discrimination*) as a proxy for understanding, they are referring specifically to reasoning-based validation as opposed to the more derivative retrieval-based one.

5.5 Validation basis is dataset dependent

We further hypothesize that validation is dataset dependent, and that models switch — for reasons currently beyond the scope of this paper — between these two modes when presented solely with a problem (without additional prompting such as Chain of Thought (Wei et al., 2022c), etc.). We expect that when a model attempts to use reasoning-based validation on a dataset by default and this form of validation significantly underperforms generation, we can improve validation accuracy up to the level of generation performance through prompting. Indeed, we find that through careful prompting (see Appendix B), we are even able to significantly improve performance even while producing only a single output token. It is possible that this prompt "switches" the model from reasoning-based to retrieval-based validation.

6 Conclusion

Whereas prior works shed light onto the performance of language models on validation tasks relative to humans for tasks with generative accuracy parity, we present a hypothesis that specifically aims to better understand the gen-val gap, the difference between generative and validation accuracies, and learnings that we may be able to use in design more holistic, consistent, and challenging benchmarks for an emerging field of reasoning frameworks. However, we recognize that further study is required to investigate the generalizability of the observations established across these 10 datasets.

7 Ethical Considerations

We do not foresee any ethical considerations in direct relation to our work. While there are broader risks from general intelligence systems, and this research contributes towards our understanding of language models and ultimately our grasp of this goal, we hope that our paper provides interpretability to language model understanding, and allows the creation of more effective datasets in measuring these emerging phenomena. We hope to further the pursuit of gradually peeling back the black box that constitute many aspects of modern large language models.

8 Limitations

A result of our choice to evaluate understanding on the most capable and popular language models, as well as our desire to access logprobabilites for up to 20 alternative next-token predictions, is that we experiment primarily on GPT-3.5 and GPT-4. We foresee potential limitations with evaluating solely on closed source language models. We also attempt to estimate the upper bound ϵ and other behaviour

through a limited number of datasets. While we attempt to choose datasets spanning a broad range of skills, choose datasets before performing any experiments, and report results on all datasets regardless of performance, we ultimately only evaluate on 10 datasets which is a fraction of the full dataset space. HotpotQA is licensed under Apache-2.0, MultispanQA: no license and publicly available by authors, 24 Game: MIT, GSM8K: MIT, MMLU: MIT, COLLIE: MIT, CSQA: no license and publicly available, Hellaswag: MIT, RACE: no license and publicly available, and PIQA: Apache-2.0. Usage of benchmarks is consistent with intended use. All benchmarks are in English, and train/test/dev splits are as originally used on each dataset. We evaluate primarily on test splits, but use validation splits where ground truth is unavailable in the test split.

Acknowledgements

Grateful for TBD.

References

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2022. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of the Association for Computational Linguistics*, 10:414–433.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439. Issue: 05.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. ArXiv:2303.12712 [cs].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. ArXiv:2110.14168 [cs].

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-Verification Reduces Hallucination in Large Language Models. ArXiv:2309.11495 [cs].

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. ArXiv:2305.14325 [cs].

Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas I. Liao, Kamilė Lukošiūtė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, Dawn Drain, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jackson Kernion, Jamie Kerr, Jared Mueller, Joshua Landau, Kamal Ndousse, Karina Nguyen, Liane Lovitt, Michael Sellitto, Nelson Elhage, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robert Lasenby, Robin Larson, Sam Ringer, Sandipan Kundu, Saurav Kadavath, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Christopher Olah, Jack Clark, Samuel R. Bowman, and Jared Kaplan. 2023. The Capacity for Moral Self-Correction in Large Language Models. ArXiv:2302.07459 [cs].

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric Xing, and Zhiting Hu. 2022. Efficient (Soft) Q-Learning for Text Generation with Limited Good Data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6969–6991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface Form Competition: Why the Highest Probability Answer Isn't Always Right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large Language Models Cannot Self-Correct Reasoning Yet. ArXiv:2310.01798 [cs].

John Schulman, Barret Zoph, and Christina Kim. 2022. Introducing ChatGPT.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. 2022. MultiSpanQA: A Dataset for Multi-Span Question Answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1260, Seattle, United States. Association for Computational Linguistics.

Xiang Lisa Li, Vaishnavi Shrivastava, Siyan Li, Tatsunori Hashimoto, and Percy Liang. 2024. Benchmarking and improving generator-validator consis-

tency of language models. In *The Twelfth International Conference on Learning Representations*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):195:1–195:35.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report. ArXiv:2303.08774 [cs].

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating llms with fewer examples.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu,

Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022c. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. 2024. The Generative AI Paradox: "What It Can Create, It May Not Understand". In *The Twelfth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R. Narasimhan. 2024. COLLIE: Systematic Construction of Constrained Text Generation Tasks. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R. Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

A Prompts used in Sub-Hypothesis 1

We use a number of prompts to encourage the language model to match or surpass generation performance, when validating answers.

Prompt P1 `Think step by step. First generate your own answer to the question and then compare this with the provided answer. Check, then double check your thinking. The last word of your response should be 'TRUE' if the answer is correct and 'FALSE' if the answer is incorrect, given the question.`

The effect of the prompts most effective at closing these gen-val gaps across our trials are shown in Figure 3.

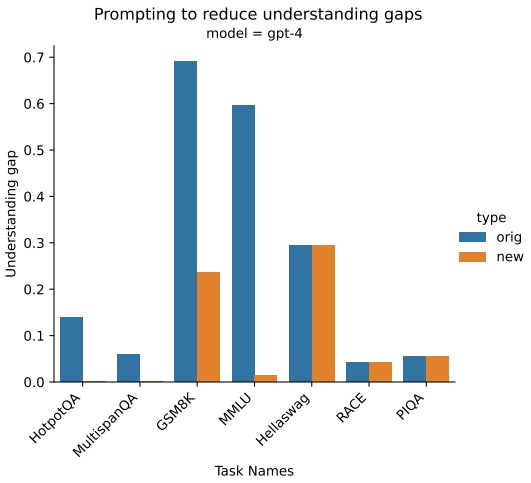

Figure 3: Reducing the gen-val gap through prompting on GPT-4

B Prompting phenomena

We notice a remarkable phenomena when attempting to investigate our subhypothesis 1 on prompting. Despite prompting the model in a typical manner to generate a well-reasoned explanation and to engage in self-reflection, the model outputs a single token `TRUE` or `FALSE` answer (see Figure 4. While we are unsure of why this happens, we note that it allows us to achieve or surpass generator performance on many tasks.

The technique performs well in practice, outperforming the classical Chain of Thought (CoT) without require additional output tokens. The GPT-4 with Chain of Thought configuration surprisingly scores 0% because instead of responding with `TRUE` or `FALSE`, the most simply responds with a numerical answer to the problem itself. We try two different prompts `You are given a question and a prediction. Answer 'TRUE' if the predicted answer is correct and 'FALSE' if the answer is incorrect. Think step by step.` and `Answer 'TRUE' if the answer is correct and 'FALSE' if the answer is incorrect. Think step by step.`. In both instances, we do not receive a single generation that contains `TRUE` or `FALSE` which is very surprising and warrants future investigation. We do note that we use the latest `gpt-4-1106-preview`, so perhaps this is the result of recent changes to the model.

	HotpotQA	MultispanQA	GSM8K
GPT-3.5	60%	**73%**	40.30%
GPT-4	66%	63%	21.50%
CoT (GPT-4)	71%	63%	0%
CnT (ours) (GPT-4)	**80%**	69%	**67%**

Table 3: Performance of Constrained Thought on GPT-4 measured against standard and Chain of Thought prompting on validation across each dataset. Note that GPT-3.5 performance actually outperforms GPT-4 on MultispanQA, despite achieving roughly similar performance in a generative mode.

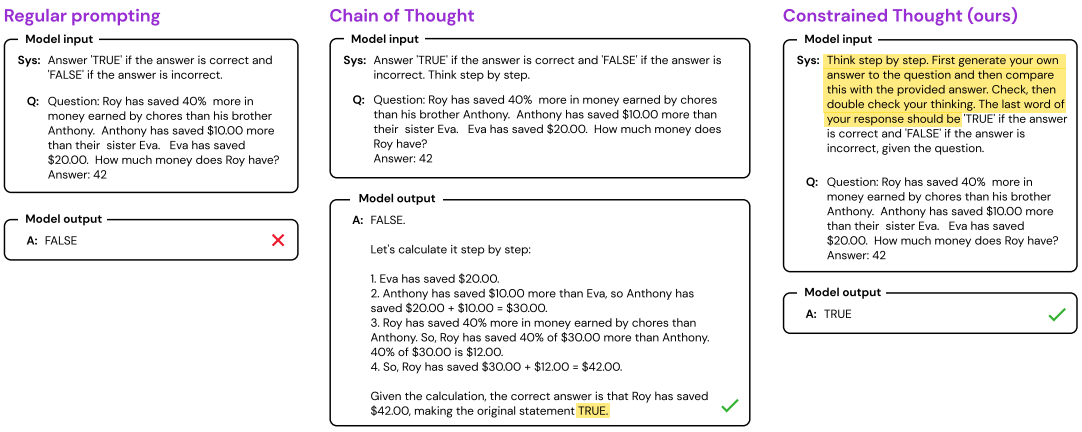

Figure 4: `Constrained Thought`, a prompting technique rivalling Chain of Thought without the additional output. As seen here, on more challenging problems, Chain of Thought may output more than one answer, which does not occur during our trials with `Constrained Thought`. The technique consists primarily of prompting the model to think and perform complex reasoning, but to then restrict the model to outputting a single character

C Additional notes

C.1 Effect of prompting on reducing the gen-val gap

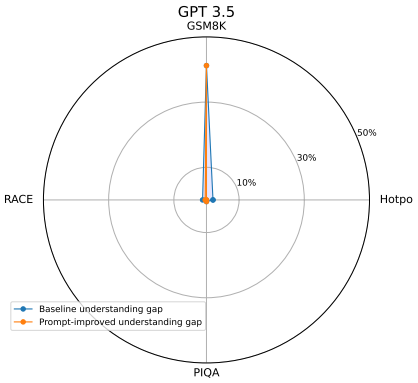

Figure 5: Lowest achieved gen-val gap by adjusting validator prompt on GPT 3.5 across various skills, compared to baseline gen-val gap

C.2 Formula used for rescaling scores

We use the following formula to rescale values in Table 2:

$$\text{rescaled score} = \left| \frac{s - r}{1 - r} \right|$$

where s is the original score, before rescaling, r is the expected score attained by a randomly-guessing model. s and r are both scores in $\{0, 1\}$. We take absolute values since models that perform worse than random guessing likely have a negative correlation. We could trivially achieve a rescaled score in practice (in a binary classification case) that is the absolute value of this negative by simply outputting the oppposite of what the model outputs.

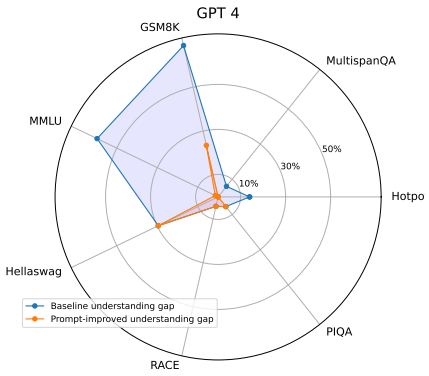

Figure 6: Lowest achieved gen-val gap by adjusting validator prompt on GPT 4 across various skills, compared to baseline gen-val gap

Systems prompts The system prompt used for generation is:

```
Generate a single token corresponding to the letter of the correct option for the following multiple choice question.
```

The system prompt used for validation is:

```
Answer 'TRUE' or 'FALSE': Is the option provided the correct answer to the question?
```

C.3 Sample Logprobs Generation

A sample generation on Hellaswag generated by GPT-4 in the Chat Completion configuration. The model generates the correct answer

System: Generate a single token corresponding to the letter of the correct option for the following multiple choice question.

User: [header]How to make fried chicken [title]Prepare the escalopes. [step]Remove the skin and tendons from the chicken. Put the chicken fillets between two sheets of plastic food wrap or parchment paper and pound with a rolling pin.

A: Scoop out medium size pieces of chicken and roll them into quarters. The approximately 2 pounds is enough to make one half pound but you'll also need about two 20 pound rolls that you can spare.

B: “The aim is to flatten them evenly to about 1 centimeter (0.4 in) (1/2 ") in depth. The flattened pieces are now known as escalopes.

C: [substeps]Do not overcook the chicken. Because there is no need to do so, you should make sure that you thoroughly slice the chicken into one or two thick slices before cutting it into four equal chunks.

D: Hold the wrap at a 45 degree angle and pound until it breaks up and falls off. Place the chicken pieces in a blender and puree.

Assistant (GPT-4): B


```
"top 20 generation, logprob pairs": {
        "B": -3.059055e-05,
        "C": -11.5781555,
        "A": -11.7656555,
        "D": -11.7969055,
        "\n": -12.2344055,
        "The": -14.4219055,
        "\n\n": -15.6406555,
        "b": -16.109406,
        "E": -16.65628,
        " B": -17.453156,
        "Incorrect": -17.81253,
        "I": -17.890656,
        " \n": -18.65628,
        "This": -18.703156,
        "**": -18.734406,
        "There": -18.828156,
        "It": -18.96878,
        "Please": -19.03128,
        "Correct": -19.078156,
        "To": -19.28128
    },
```


C.4 Sample Logprobs Validation

A sample generation on MMLU generated by GPT-4 in the Chat Completion configuration. The model mislabels both options A and D as TRUE, whereas the correct answer is A. In this example we only see the label on A. Notice that the model is fairly uncertain about the TRUE/FALSE label given the similar logprobs, something that the softmax metric uniquely captures.

System: Answer ‘TRUE‘ or ‘FALSE‘: Is the option provided the correct answer to the question?

User: In a medical class of 147 students, the distribution of scores on a biochemistry final examination has a mean equal to 67, a median equal to 76, a mode equal to 80, a standard deviation equal to 5.5, and a variance equal to 30.25. Three students were unable to take the test on the scheduled date and were given a different form of the exam 1 week later. Which parameter is most likely to be the least biased estimator of central tendency for this distribution of biochemistry test scores? Is the following option correct? Option: Mean

Assistant: TRUE (with logprobs 0.58), FALSE (with logprob 0.41)

The above is run with each option of an example in the dataset.