The Good, The Bad, and The Greedy: Evaluation of LLMs Should Not Ignore Non-Determinism

Yifan Song♡ **Guoyin Wang** **Sujian Li**♡ **Bill Yuchen Lin**♣♠

♡Peking University ♣Univesity of Washington ♠Allen Institute for AI

yfsong@pku.edu.cn byuchen@uw.edu

Abstract

Current evaluations of large language models (LLMs) often overlook non-determinism, typically focusing on a single output per example. This limits our understanding of LLM performance variability in real-world applications. Our study addresses this issue by exploring key questions about the performance differences between greedy decoding and sampling, identifying benchmarks' consistency regarding non-determinism, and examining unique model behaviors. Through extensive experiments, we observe that greedy decoding generally outperforms sampling methods for most evaluated tasks. We also observe consistent performance across different LLM sizes and alignment methods, noting that alignment can reduce sampling variance. Moreover, our best-of-N sampling approach demonstrates that smaller LLMs can match or surpass larger models such as GPT-4-Turbo, highlighting the untapped potential of smaller LLMs. This research shows the importance of considering non-determinism in LLM evaluations and provides insights for future LLM development and evaluation. [1]

1 Introduction

When evaluating a large language model (LLM), two common generation configurations are commonly used: greedy decoding and nucleus sampling (Holtzman et al., 2020). It's important to note that given a particular input, the same LLM may generate significantly different outputs under various decoding configurations, a phenomenon known as non-determinism in generation. However, most evaluations of LLMs are based on a single output per example. This practice is primarily due to practical considerations, as LLM inference and evaluation can be computationally expensive. Neglecting non-determinism in generation significantly limits our comprehensive understanding of LLMs. Additionally, without reporting the standard deviation in most current LLM evaluations, it is difficult to measure the variability and dynamics of LLMs in real-world applications.

For certain capabilities such as math reasoning (Cobbe et al., 2021; Hendrycks et al., 2021b) and coding, greedy generation is preferred to ensure fair comparisons. Nonetheless, it remains unclear whether there are significant differences in performance between greedy decoding and sampling. Recent investigations have also highlighted potential issues of instability in LLMs (Li et al., 2024a; Hassid et al., 2024). In a study where the best answer was selected from 256 random generations, the Llama-2-7B model achieved an impressive 97.7% accuracy in solving GSM8K questions, even surpassing GPT-4 (Li et al., 2024a). This phenomenon further underscores the enormous potential of LLMs in their non-deterministic outputs.

Previous studies (Sclar et al., 2023; Mizrahi et al., 2024; Alzahrani et al., 2024) have extensively examined the influence of different sources of variance on LLM generation, including prompts and in-context examples. However, the impact of different decoding configurations on LLM performance has not been fully explored. Herein, we aim to investigate a series of critical questions regarding the non-determinism of LLM generations:

- **Q1**: *How does the performance gap between greedy decoding and sampling differ?*
- **Q2**: *When is greedy decoding better than sampling, and vice versa? Why?*
- **Q3**: *Which benchmark is most/least consistent with respect to non-determinism?*
- **Q4**: *Do any models possess unique patterns?*

Apart from Q1-Q4 in Sec. 3, we also explore the ***scaling*** effect on non-determinism (Sec. 4.1), the ***alignment*** effect on non-determinism (Sec. 4.2), and how ***temperature*** and ***repetition*** influence on generation (Sec. 4.3, 4.4).

[1] Code and data are available at https://github.com/Yifan-Song793/GoodBadGreedy

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4195–4206
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

Our extensive results reveal these findings:

- For most benchmarks we evaluated, a notable performance gap is observed between greedy generation and the average score of multiple sampling. In certain cases, the performance ranking under different generation configurations differs.
- Greedy decoding exhibits superior performance than sampling methods on most evaluated benchmarks, except for AlpacaEval where sampling shows higher win rate.
- LLMs displayed consistent performance across different generation configurations for benchmarks with constrained output spaces, such as MMLU and MixEval. Notably, tasks involving math reasoning and code generation were most impacted by sampling variance.
- The above findings remain consistent across different sizes and families of LLMs.
- Alignment methods, e.g., DPO (Rafailov et al., 2024), can significantly reduce the sampling variance for most benchmarks.
- High temperature will significantly harm the reasoning and code generation capabilities of LLMs, while higher repetition penalty leads to improved performance on AlpacaEval.

Given the non-deterministic nature of LLM generation, it is essential to explore how to leverage this characteristic. In Sec. 5, we enable LLMs to generate multiple responses for a task and pick the optimal answer with off-the-shelf reward models. In this best-of-N sampling setting, we observe that 8B-level LMs exhibit the potential to surpass GPT-4-Turbo on several benchmarks. These findings underscore the importance of scaling inference time compute through repeated sampling (Snell et al., 2024; Brown et al., 2024) for further enhancing the performance of existing LLMs.

2 Experimental Setup

Benchmarks. We select multiple benchmarks for our experiments, encompassing abilities of general instruction-following, knowledge, math reasoning, coding, etc. As summarized in Table 1, the selected benchmarks are: AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024b), WildBench v2 (Lin et al., 2024), MixEval (Ni et al., 2024), MMLU-Redux (Gema et al., 2024), GSM8K (Cobbe et al., 2021), and HumanEval (Chen et al., 2021).

Dataset	Instance Num.	Sample Num.	Metric
AlpacaEval 2	805	16	LC
Arena-Hard	500	16	WR
MixEval	4000	16	Score
WildBench v2	1024	16	WB-Score
MMLU-Redux	3000	32	Acc
GSM8K	1319	128	EM
HumanEval	164	128	Pass@1

Table 1: Statistics of datasets.

AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024b) and WildBench v2 (Lin et al., 2024) are general instruction-following benchmarks. AlpacaEval consists of 805 questions, Arena-Hard incorporating 500 well-defined technical problem-solving queries. WildBench including 1024 challenging real users tasks, which are categorized into 12 categories to enable a fine-grained analysis of LLM capabilities. For AlpacaEval 2, we report the length-controlled win rate (LC). For Arena-Hard, we report the win rate (WR) against the baseline model. For WildBench, we use task-wise scores and the corresponding task-macro WB-Score (Lin et al., 2024) as the metrics.

MixEval (Ni et al., 2024) is a comprehensive mixture of various off-the-shelf ground-truth style benchmarks. Since the original MMLU (Hendrycks et al., 2021a) benchmark is huge and contain numerous ground truth errors (Wang et al., 2024b; Gema et al., 2024), we use MMLU-Redux (Gema et al., 2024) which is a subset of 3000 manually re-annotated questions across 30 MMLU subjects. We also include GSM8K (Cobbe et al., 2021), and HumanEval (Chen et al., 2021), two popular benchmarks for evaluating the math and code generation abilities of LLMs.

LLMs. We test several open-weight LLMs, including Llama-3-Instruct (Meta, 2024), Yi-1.5-Chat (Young et al., 2024), Qwen-2-Instruct (Bai et al., 2023), Mistral (Jiang et al., 2023a), which are widely used. A proprietary LLM, GPT-4-Turbo, is included for comparison. We also consider models of different sizes in the same family such as Qwen2 and Yi-1.5 for more analysis. To study the effect of alignment techniques, we evaluate models trained with different alignment methods, including DPO (Rafailov et al., 2024), KTO (Ethayarajh et al., 2024), IPO (Azar et al., 2024), ORPO (Hong et al., 2024), RDPO (Park et al., 2024), and SimPO (Meng et al., 2024). We use the checkpoints released by Meng et al. (2024).

Setup. We aim to compare the performance of LLMs under different decoding configurations. We

Model	AlpacaEval 2 (N=16)				Arena-Hard (N=16)				MixEval (N=16)			
	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ
GPT-4-Turbo	49.6	50.1	0.76	2.5	80.1	75.2	**1.31**	3.6	89.2	88.8	0.18	0.8
Llama-3-8B-Instruct	26.8	29.2	0.88	2.8	23.5	18.4	0.71	2.7	74.6	72.5	0.25	0.9
Yi-1.5-6B-Chat	17.5	18.0	0.91	3.4	13.7	11.8	0.88	3.1	70.0	68.6	0.26	1.0
Yi-1.5-9B-Chat	23.1	24.1	0.91	3.4	32.8	27.0	1.25	4.4	74.0	72.7	**0.35**	1.4
Yi-1.5-34B-Chat	34.9	35.0	0.99	3.9	42.8	40.9	1.82	5.7	81.9	81.8	0.47	1.5
Qwen2-7B-Instruct	18.2	19.1	**2.51**	8.6	23.7	16.1	0.87	3.1	76.2	76.2	0.21	0.6
Mistral-7B-Instruct-v0.2	15.4	13.0	1.02	4.2	12.5	12.6	0.57	2.0	69.8	70.0	0.24	0.9

Model	MMLU-Redux (N=32)				GSM8K (N=128)				HumanEval (N=128)			
	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ
GPT-4-Turbo	82.6	82.4	0.43	1.6	84.5	83.8	0.77	2.5	89.6	84.1	2.65	11.0
Llama-3-8B-Instruct	47.8	50.7	**0.70**	2.8	67.6	64.4	**2.50**	13.4	58.5	31.8	3.62	18.3
Yi-1.5-6B-Chat	52.1	49.6	0.67	2.5	74.5	73.1	0.92	4.1	48.2	35.7	4.86	19.5
Yi-1.5-9B-Chat	65.5	64.3	0.53	2.3	82.9	81.0	0.69	3.9	55.5	36.4	**4.92**	27.5
Yi-1.5-34B-Chat	83.2	82.2	0.34	1.1	85.4	81.7	0.56	2.9	64.6	49.3	4.08	21.4
Qwen2-7B-Instruct	64.4	61.7	0.46	2.1	83.5	72.0	1.74	11.3	67.7	48.2	4.68	27.4
Mistral-7B-Instruct-v0.2	49.7	48.4	0.49	2.2	45.9	42.0	0.99	5.1	37.8	25.9	2.52	14.0

Table 2: Results on six popular benchmarks. “Sample” and “Std.” denotes the average score and the standard deviation of “N” runs under sampling setup. “Δ” denotes the performance gap between the best and worst run. Scores where greedy decoding surpasses the sampling average are highlighted in green, while those lower are marked in red. The intensity of the color indicates the magnitude of the difference (best viewed in color).

select greedy decoding and sampling generation for the main comparison. For sampling, we set the temperature to 1.0 and top-p to 1.0.

We use official evaluation scripts for AlpacaEval 2, Arena-Hard, WildBench, and MixEval. For MMLU-Redux, instead of using the next token probability of the choice letters, we employ zero-shot CoT and encourage the model to generate the answer in the form of natural language sentence. For GSM8K and HumanEval, we use Open-Instruct framework (Wang et al., 2023) to evaluate the models, which may differ from zero-shot CoT. We will run more comprehensive evaluations on these two benchmarks in the future. We sample 16 completions for AlpacaEval 2, Arena-Hard, WildBench, and MixEval, 32 completions for MMLU-Redux, 128 for GSM8K and HumanEval.

3 Experimental Results

We present our experiment results in Table 2 and Table 3. We analyze the results and answer several important research questions around the non-determinism of LLM generations as follows.

Q1. How does the performance gap between greedy decoding and sampling differ?

From the results, we observe a consistent performance gap between greedy decoding and the sampling method (significance test in Appendix A). This disparity holds true across various LLMs, whether they are proprietary or open-source, and across multiple benchmarks encompassing instruction-following, language understanding, math reasoning, and code generation. For WildBench, which enables fine-grained analysis of LLM capabilities, the performance gap is also evident across all task categories, as shown in Table 3.

Q2. When is greedy decoding better than sampling, and vice versa? Why?

For most evaluated tasks and models, greedy decoding outperforms sampling. However, AlpacaEval serves as a notable exception, where sampling demonstrates superior performance.

GSM8K and HumanEval are reasoning tasks requiring LLMs to solve specific math or coding problems with definite solutions. MixEval also follows a deterministic pattern with its ground-truth-based benchmarks. While AlpacaEval, Arena-Hard, and WildBench are open-ended instruction-following benchmarks, AlpacaEval exhibits a contrasting behavior compared to the others. The potential reasons are two folds: Firstly, the task category distributions vary across different benchmarks. As highlighted by Lin et al. (2024), 50% of instances in AlpacaEval are information-seeking, whereas more than 50% in Arena-Hard are related to coding and debugging. Furthermore, the difficulty of instances might play an important role. The tasks in both Arena-Hard and WildBench, sourced

Metric	Llama-3-8B-Instruct				Yi-1.5-6B-Chat				Qwen2-7B-Instruct			
	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ	Greedy	Sample	Std.	Δ
WB-Score	29.6	26.2	1.65	5.7	23.9	22.4	1.67	5.3	32.7	23.8	2.13	7.7
Creative Tasks	42.2	42.4	1.77	6.7	32.1	32.1	2.33	10.3	39.6	31.4	2.21	8.5
Planning & Reasoning	33.8	31.4	1.19	3.6	27.9	27.4	1.77	5.7	36.0	28.1	1.95	6.1
Math & Data Analysis	17.8	16.0	2.85	9.2	17.4	17.5	1.99	6.5	27.6	18.5	2.69	10.4
Info/Advice Seeking	39.0	37.4	1.30	5.5	32.5	30.2	1.80	6.3	40.3	32.2	1.84	6.5
Coding & Debugging	24.1	16.0	3.12	10.9	16.7	12.8	1.70	5.4	26.3	15.5	2.82	9.3

Table 3: Results on **WildBench v2**, with sampling N=16 generations for each model. In addition to WB-Score, we also report the score for each task category.

Benchmark	Kendall's τ	p-value
AlpacaEval 2	0.916	1.4E-5
Arena-Hard	0.872	1.8E-6
MixEval	0.789	2.3E-4
MMLU-Redux	0.872	1.8E-6
GSM8K	0.714	0.030
HumanEval	0.778	0.002

Table 4: Kendall's τ correlation and the p-value for the hypothesis test (the null hypothesis is $\tau = 0$) for LLM performance rankings of greedy decoding and average score of multiple samplings on six benchmarks.

from real users, pose substantial challenges. On the other hand, instances in AlpacaEval are comparatively simpler.

In summary: 1) Greedy decoding generally proves more effective for most tasks. 2) In the case of AlpacaEval, which comprises relatively simpler open-ended creative tasks, sampling tends to generate better responses.

Q3. Which benchmark is most/least consistent with respect to non-determinism?

In terms of the performance gap between two decoding configurations and the standard deviation across different samplings, MixEval and MMLU exhibit the highest stability. This stability can be attributed to the constrained answer space of these benchmarks. Specifically, MMLU is structured in a multiple-choice format, and MixEval, comprising various ground-truth-based benchmarks, prompts LLMs to generate short answers, further limiting the output space.

In contrast, GSM8K and HumanEval are relatively less stable with respect to non-deterministic generations. The performance gap between the best and worst samplings can exceed 10.0 points. To address this instability, the LLM community has adopted specific evaluation protocols. For GSM8K, all models are evaluated using greedy decoding. For HumanEval, models are sampled multiple times, and Pass@k is used as the final metric to ensure reliable comparison.

Different decoding configurations can even alter the model rankings in some cases. For example, on Arena-Hard, Qwen2-7B is slightly better than Llama-3-8B when both use greedy decoding; However, Llama-3-8B may outperform Qwen2-7B when both decode by sampling. To measure the change in ranking induced by non-determinism, we compute Kendall's τ (Kendall, 1938): $\mathbb{N}^n \times \mathbb{N}^n \to [-1, 1]$, a non-parametric statistic which measures the correspondence between two rankings R_1, R_2 (greedy decoding and average score of multiple samplings, in our case). τ is formally defined as:

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + T) \cdot (P + Q + U)}}, \quad (1)$$

where P is the number of concordant pairs, Q is the number of discordant pairs, T the number of ties only in R_1, and U the number of ties only in R_2. Therefore, $\tau > 0$ indicates that most pairs are concordant, and $\tau < 0$ indicates that most pairs are discordant. As shown in Table 4, the ranking in GSM8K and HumanEval are less robust against different decoding settings.

Q4. Do the models possess distinctive characteristics?

GPT-4-Turbo shows consistent performance across multiple tasks, with a smaller performance gap between greedy decoding and sampling, as well as improved sampling quality. Some open-weight LLMs, however, exhibit unique characteristics. For example, Mistral-7B-Instruct-v0.2 displays inverse behavior on open-ended instruction following tasks like AlpacaEval and Arena-Hard

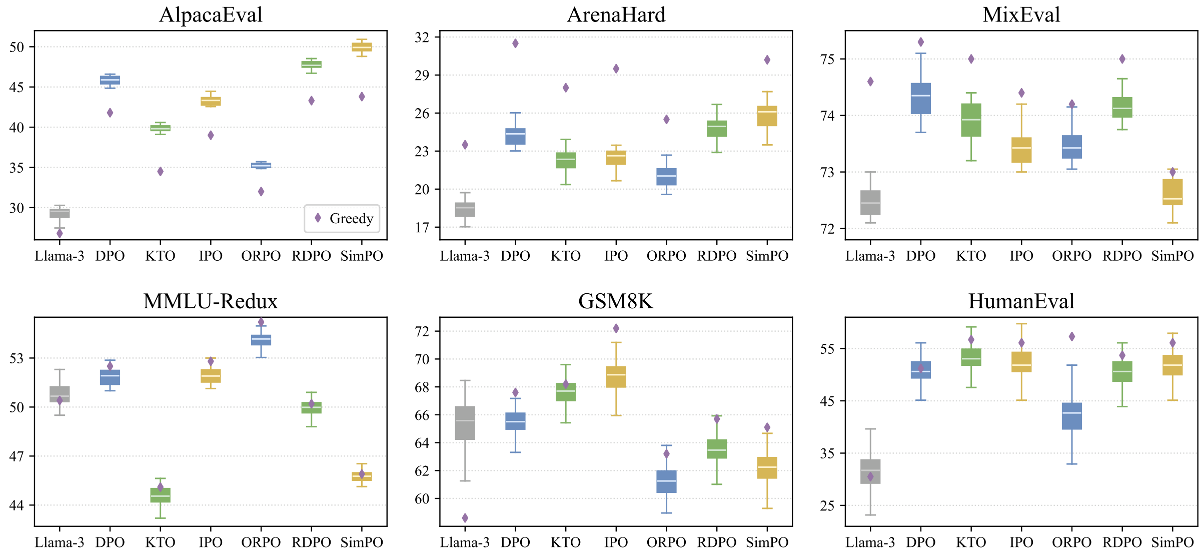


Figure 1: Alignment effects on non-determinism.

when compared to other models. Similarly, Llama-3-8B-Instruct performs better by sampling than by greedy decoding on MMLU, which is unlike the behavior of other models.

These observations raise intriguing questions for future research. Why do certain models exhibit inverse behavior on specific tasks? Can these unique characteristics be leveraged to develop more robust LLMs? These questions highlight the need for deeper explorations into the underlying mechanisms of LLMs. Such research could significantly enhance our understanding of how different models and training impact model behavior.

4 How Various Factors Influence Non-Determinism?

In this section, we further investigate how various factors, such as scaling, alignment, and several decoding parameters, influence non-determinism.

4.1 Scaling Effect on Non-Determinism

Some might assume that larger LMs will have lower uncertainty in decoding, leading to lower variance in performance when sampling. However, our results challenge this assumption.

We use the Yi-1.5-Chat and Qwen2-Instruct series to investigate the scaling effect. The results for the Yi-1.5 and Qwen2 series are presented in Table 2 and Table 5, respectively. Performance differences are observed across LLMs of various sizes, ranging from 0.5B to 34B parameters. The findings in Section 3 are consistent across different model sizes. However, no pattern related to the number of model parameters could be identified. For instance, scaling parameters does not result in lower sampling variance. Notably, Qwen2-7B-Instruct shows higher variance on AlpacaEval and HumanEval compared to its smaller counterparts.

Model	**AlpacaEval**			**MMLU**		
	G	S	Std.	G	S	Std.
Qwen2-0.5B-Instruct	1.1	1.7	0.77	36.4	37.0	0.70
Qwen2-1.5B-Instruct	1.9	3.3	0.88	42.6	42.1	0.68
Qwen2-7B-Instruct	18.2	19.1	2.51	61.0	61.7	0.46
Model	**GSM8K**			**HumanEval**		
	G	S	Std.	G	S	Std.
Qwen2-0.5B-Instruct	31.7	14.3	1.86	28.0	10.8	2.14
Qwen2-1.5B-Instruct	63.1	36.5	3.20	40.9	22.6	2.94
Qwen2-7B-Instruct	83.5	72.0	1.74	67.7	48.2	4.68

Table 5: Evaluation results on Qwen2-Instruct with different model sizes.

4.2 Alignment Effect on Non-Determinism

Alignment methods, such as DPO, enhance LLMs by learning from preference data. We evaluate the effects of alignment methods such as DPO, KTO, and SimPO, using Llama-3-8B-Instruct as the training starting point (Meng et al., 2024).

As shown in Figure 1, after applying these methods, both greedy decoding and sampling performances are affected. In several tasks, including AlpacaEval, MMLU, GSM8K, and HumanEval, a decrease in standard deviation is observed, suggesting that alignment may reduce the diversity of sampling outputs. However, it is crucial to note that not all alignment methods consistently improve model

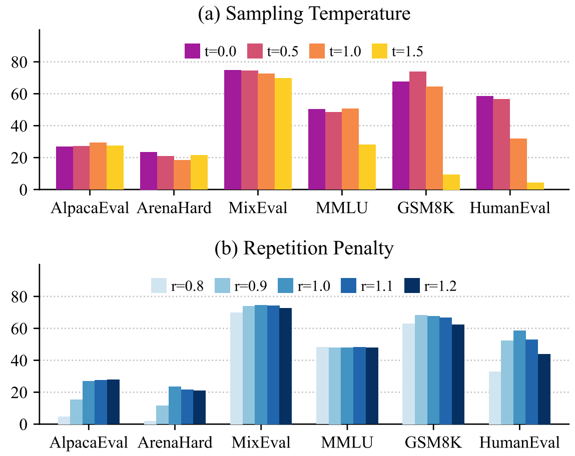


Figure 2: (a) Temperature effects on non-determinism. (b) Repetition penalty effects on generation. We compare performance of Llama-3-8B-Instruct with different generation parameters.

performance. For instance, KTO and SimPO lead to a performance decline in MMLU. Furthermore, SimPO's effectiveness appears limited on the recently introduced MixEval benchmark.

4.3 Temperature Effect on Non-Determinism

For sampling generation, temperature serves as a control mechanism for the randomness of the sampling process, where lower values make the model more deterministic, whereas higher values make the model more random. In this section, we present an ablation study to evaluate the effect of varying temperatures on non-determinism generation.

As depicted in Figure 2(a), we observe that, for AlpacaEval, higher temperature will lead to slightly better performance, which aligns with the results in Sec. 3. A recent study (Renze and Guven, 2024) finds that, on multiple-choice QA tasks, changes in temperature from 0.0 to 1.0 do not have a statistically significant impact on LLM performance. Our results on MMLU aligns with their findings. Another findings emerges when the temperature is extremely high, such as 1.5. Comparing with open-ended instruction following, a high temperature significantly impacts the reasoning and code generation capabilities of LLMs and the model struggles to solve questions in GSM8K and HumanEval. However, it still manages to perform relatively well in open-ended instruction following tasks, such as AlpacaEval and ArenaHard.

4.4 Repetition Effect on Generation

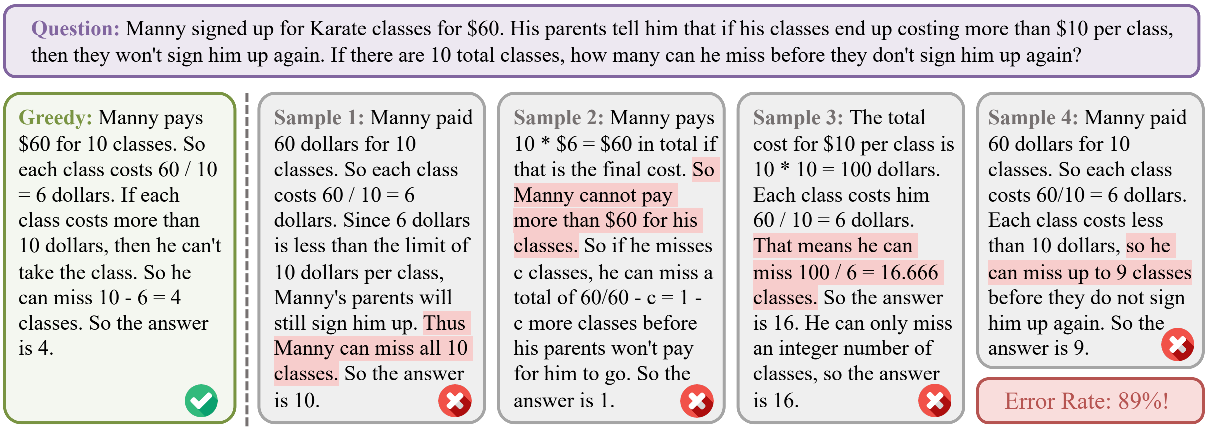
In addition to parameters that control greedy search and sampling, there are other parameters that influence the generation process, such as the repetition penalty (Keskar et al., 2019). Here we examine the effect of repetition penalty on generation. Repetition penalty penalizes new tokens based on whether they appear in the prompt and the generated text so far. Values over 1.0 encourage the model to use new tokens, while values under 1.0 promote the reuse of tokens. The default repetition penalty in generation is set at 1.0.

As illustrated in Figure 2(b), in most cases, it is advisable not to adjust this parameter, as maintaining the default value tends to yield the best performance. For AlpacaEval, a higher repetition penalty like 1.2 results in marginally improved performance. This improvement may be linked to GPT judges' preference for shorter, more concise answers. Regarding MixEval and MMLU, repetition penalty has a minimal impact on the model's performance, since both benchmarks advocate for the model to generate concise answers. Interestingly, for GSM8K, the model achieves the best performance when the repetition penalty is set at 0.9, and increasing this penalty parameter will cause a performance decline. This phenomenon can be attributed to the nature of mathematical reasoning, which frequently necessitates the repetition of numbers and conditions outlined in the question.

4.5 Surface Patterns in Non-Determinism Generation?

We try to explore the surface patterns in non-determinism generation. Firstly, we compare the generation length of different generation configurations in Table 6. The generation length for AlpacaEval and ArenaHard is defined as the length of the model's response, while for MMLU and GSM8K, it refers to the length of the final answer with chain-of-thoughts. We observe that the completions generated by greedy decoding are typically marginally shorter than those produced via sampling generation. However, this pattern deviates in the case of Yi series models on AlpacaEval and GSM8K, where the lengths of responses produced by both greedy decoding and sampling methods are comparable.

We also take Qwen2-7B-Instruct on GSM8K as a case study, where the greedy decoding significantly outperforms the sampling generation (83.5 vs. 72.0). As depicted in Figure 3, greedy decoding



Question: Manny signed up for Karate classes for $60. His parents tell him that if his classes end up costing more than $10 per class, then they won't sign him up again. If there are 10 total classes, how many can he miss before they don't sign him up again?

Greedy: Manny pays $60 for 10 classes. So each class costs 60 / 10 = 6 dollars. If each class costs more than 10 dollars, then he can't take the class. So he can miss 10 - 6 = 4 classes. So the answer is 4. ✓

Sample 1: Manny paid 60 dollars for 10 classes. So each class costs 60 / 10 = 6 dollars. Since 6 dollars is less than the limit of 10 dollars per class, Manny's parents will still sign him up. Thus Manny can miss all 10 classes. So the answer is 10. ✗

Sample 2: Manny pays 10 * $6 = $60 in total if that is the final cost. So Manny cannot pay more than $60 for his classes. So if he misses c classes, he can miss a total of 60/60 - c = 1 - c more classes before his parents won't pay for him to go. So the answer is 1. ✗

Sample 3: The total cost for $10 per class is 10 * 10 = 100 dollars. Each class costs him 60 / 10 = 6 dollars. That means he can miss 100 / 6 = 16.666 classes. So the answer is 16. He can only miss an integer number of classes, so the answer is 16. ✗

Sample 4: Manny paid 60 dollars for 10 classes. So each class costs 60/10 = 6 dollars. Each class costs less than 10 dollars, so he can miss up to 9 classes before they do not sign him up again. So the answer is 9. ✗

Error Rate: 89%!

Figure 3: Case Study on non-determinism: Qwen2-7B-Instruct on GSM8K. Greedy decoding can effectively address the question. However, in 128 sampling generations for the same question, the error rate is 89%.

Model	AlpacaEval		ArenaHard	
	Len-G	Len-S	Len-G	Len-S
GPT-4-Turbo	377	389	629	641
Llama-3-8B-Instruct	417	435	589	570
Yi-1.5-6B-Chat	477	479	670	636
Yi-1.5-9B-Chat	500	502	672	692
Yi-1.5-34B-Chat	450	453	693	705
Qwen2-7B-Instruct	420	410	573	594
Mistral-7B-Instruct-v0.2	323	372	533	550
Model	**MMLU**		**GSM8K**	
	Len-G	Len-S	Len-G	Len-S
GPT-4-Turbo	257	272	149	150
Llama-3-8B-Instruct	130	128	65	94
Yi-1.5-6B-Chat	145	158	127	132
Yi-1.5-9B-Chat	160	172	138	140
Yi-1.5-34B-Chat	263	272	143	142
Qwen2-7B-Instruct	75	90	121	139
Mistral-7B-Instruct-v0.2	135	144	121	135

Table 6: Length comparison. Cases where greedy decoding generates shorter responses than sampling average are highlighted in blue, and marked in purple vice versa.

solves the question effectively. Nonetheless, when it is the turn for sampling generation, the error rate surges to 89% within 128 responses. This observation suggests that the sampling method could potentially harm reasoning capabilities for LLMs.

5 What is the Full Potential of Non-Determinism?

Current evaluations of LLMs mainly assess them based on a single output per instance, which limits our understanding of their full potential. In this section, we focus on answering the question: if an LLM is allowed to try multiple times, how much can it improve its performance on a challenge task? In other words, we are assessing the performance of scaling LLM inference time compute.

Following Jiang et al. (2023b) and Li et al. (2024a), we adopt a Best-of-N setting, which samples more than one completions from a weak LLM and then selects the best answer from N sampled responses. To accomplish this, we employ off-the-shelf reward models, such as ArmoRM (Wang et al., 2024a) and FsfairX (Xiong et al., 2024a), to rank the responses of Llama-3-8B-Instruct, selecting the one with the highest reward. We also include an "oracle" baseline which directly picks the best response as the upper bound of best-of-N strategy.

The results are depicted in Figure 4. We observe a significant performance enhancement when applying simple best-of-N strategy for multiple sampled responses. Notably, with the oracle selection, **even smaller LLMs like Llama-3-8B-Instruct can outperform GPT-4-Turbo on MMLU, GSM8K, and HumanEval**. This finding underscores that compact-sized LLMs already exhibit robust capabilities, highlighting that a more significant challenge in alignment is to robustly decode such knowledge and reasoning paths. Furthermore, cutting-edge reward models can also select superior responses from multiple generations, and can outperform GPT-4-Turbo on GSM8K with only 8 samples. However, there is still a huge performance gap between reward models and the oracle baseline, indicating ample room for improvement. Simply scaling the sample numbers also fails to further improve the performance of best-of-N.

Building upon these promising findings, there are two ways to further enhance the performance of smaller LLMs. Firstly, probability calibration techniques can guide LLMs towards generating

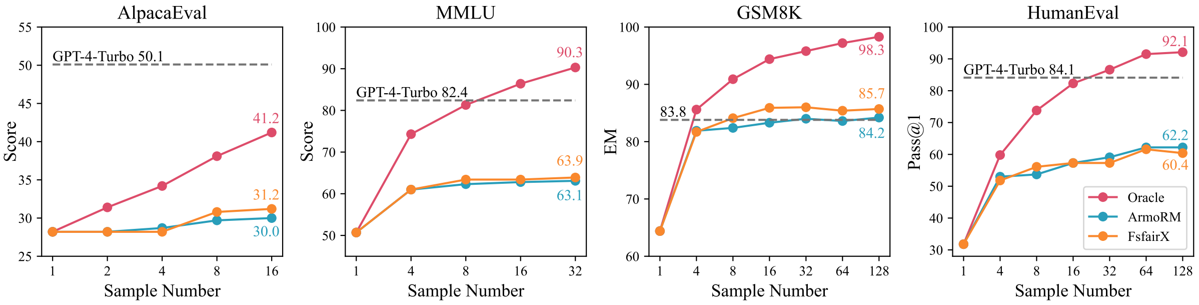


Figure 4: Potential of Llama-3-8B-Instruct. We use the setting of "Best-of-N", which selects the best response from N outputs for each example. We employ off-the-shelf reward models to rank the responses and select the one with the highest reward, while "Oracle" means the upper bound of Best-of-N method.

superior answers with higher likelihoods. Alignment methods, specifically preference optimization (Rafailov et al., 2024), play a pivotal role in this process. Secondly, strategies for identifying better samples out of many generations warrant attention. Reward modeling for re-ranking and fusing multiple outputs is thus a key direction (Jiang et al., 2023b). Self-consistency (Wang et al., 2022) and advanced prompting techniques (Yao et al., 2023; Lin et al., 2023), which employs heuristic selection from multiple completions, is also worth further exploration. It is worthy to noting, Monte Carlo Tree Search (Sutton, 2018), an advanced sampling method, has been employed in solving math reasoning (Chen et al., 2024; Zhang et al., 2024a; Luo et al., 2024) and LLM agent tasks (Zhang et al., 2024b; Xiong et al., 2024b) to enhance the performance of the policy model.

6 Related Work

LLM Evaluation In recent years, the development of various benchmarks has significantly advanced the evaluation of LLMs. Benchmarks like MMLU (Hendrycks et al., 2021a), HellaSwag (Zellers et al., 2019), and ARC (Clark et al., 2018) have expanded the scope by assessing capabilities across knowledge understanding, and complex reasoning. AlpacaEval (Li et al., 2023), ArenaHard (Li et al., 2024b), and WildBench (Lin et al., 2024), leveraging frontier models as judges, evaluate open-ended instruction-following capabilities. Moreover, GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), HumanEval (Chen et al., 2021) focus on evaluating math reasoning and code generation capabilities. MixEval (Ni et al., 2024), consisting of several off-the-shelf benchmarks, serves as a reliable and efficient LLM evaluation.

Due to the costly nature of LLM inference and evaluation process, most evaluations of LLMs rely on a single output per example. In this paper, we aim to explore the impact of various generation configurations, particularly non-deterministic generations, on the performance of LLMs.

Decoding Strategy Given a prompt, LLMs rely on a decoding strategy to auto-regressively generate response. The simplest decoding method, greedy decoding, selects the next token with the highest probability. Beam search (Freitag and Al-Onaizan, 2017), an improved version of greedy search, retains the top-B tokens with the highest probability at each time step. In order to generate diverse responses, non-determinism generation methods, such as Top-k (Fan et al., 2018) and Top-p sampling (Holtzman et al., 2020), randomly picks the next token based on the probability distribution. The temperature parameter serves to balance response quality and diversity (Ackley et al., 1985). Other decoding parameters, like length and repetition penalties (Keskar et al., 2019), are also available to further control the generation process.

7 Conclusion & Future directions

We investigate a series of critical yet overlooked questions around non-determinism of LLM generations. After evaluating several LLMs across seven commonly used benchmarks, we have answered several intriguing research questions. Further analysis also provides insights on how scaling and alignment will effect on non-determinism generation. We hope this work can enhance our comprehension of the generation methods and the widely used benchmarks. Our evaluation results can also be used for improving future research. For example, our best-of-N results can serve as a benchmark for assessing reward models (Lambert et al., 2024).

Limitations

The comparison of greedy decoding and sampling in this work reveals intriguing findings. However, it is crucial to acknowledge the limitations of our research. 1) Our evaluation exclusively relies on off-the-shelf benchmarks, neglecting the analysis of other content characteristics such as language style. 2) We observe that in most scenarios, greedy decoding will generate better responses than sampling. However, the underlying principle behind this phenomenon remains unknown. 3) While we showcase the remarkable potential of LLMs to exhibit robust capabilities, how to incorporate methods, such as self-consistency and MCTS, to improve the performance of LLMs in a multiple generation setting is under-explore.

Ethics Statement

This work fully complies with the Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169.

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yusef Almushaykeh, Faisal Mirza, Nouf Alotaibi, Nora Altwairesh, Areeb Alowisheq, et al. 2024. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. *arXiv preprint arXiv:2402.01781*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *ArXiv preprint*, abs/2309.16609.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv preprint*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *ArXiv preprint*, abs/2402.01306.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. 2024. Are we done with mmlu? *ArXiv preprint*, abs/2406.04127.

Michael Hassid, Tal Remez, Jonas Gehring, Roy Schwartz, and Yossi Adi. 2024. The larger the better? improved llm code-generation via budget reallocation. *ArXiv preprint*, abs/2404.00725.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *ArXiv preprint*, abs/2103.03874.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *ArXiv preprint*, abs/2403.07691.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *ArXiv preprint*, abs/2310.06825.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Annual Meeting of the Association for Computational Linguistics*.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv preprint*, abs/1909.05858.

Nathan Lambert, Valentina Pyatkin, Jacob Daniel Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hanna Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. *ArXiv preprint*, abs/2403.13787.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024a. Common 7b language models already possess strong math capabilities. *ArXiv preprint*, abs/2403.04706.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024b. From live data to high-quality benchmarks: The arena-hard pipeline.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *ArXiv preprint*, abs/2406.04770.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Prithviraj Ammanabrolu, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *ArXiv preprint*, abs/2305.17390.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *ArXiv preprint*, abs/2405.14734.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. 2024. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures. *arXiv preprint arXiv:2406.06565*.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. *ArXiv preprint*, abs/2403.19159.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *ArXiv preprint*, abs/2402.05201.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book*.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024a. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *ArXiv preprint*, abs/2406.12845.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv preprint*, abs/2203.11171.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *ArXiv preprint*, abs/2406.01574.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. 2024a. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *Preprint*, arXiv:2312.11456.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024b. Watch every step! llm agent learning via iterative step-level process refinement. *arXiv preprint arXiv:2406.11176*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv preprint*, abs/2305.10601.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *ArXiv preprint*, abs/2403.04652.

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.

Di Zhang, Jiatong Li, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024a. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2024b. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *arXiv preprint arXiv:2408.15978*.

A Significance Test

We use one-sample t-test to determine whether the mean performance of multiple sampling is significantly different from the score of greedy decoding. As shown in Table 8, for most cases, p value is less than 0.05, supporting our findings in Section 3.

B Computational Cost of Best-of-N

In Section 5, we find that . Despite the better performance, best-of-N sampling will introduce extra computational overhead. Here we provide an analysis of these costs. Specifically, we evaluated Llama-3-8B-Instruct on GSM8K using an A100 80G GPU with vLLM and used FsfairX as the reward model. Since the reward model can be deployed in parallel, our focus is on the cost associated with generating multiple outputs during sampling.

Table 7 illustrates that best-of-N sampling yields significant performance improvements at a reasonable computational overhead. For instance, with a 1.26x increase in computational cost, the accuracy of the smaller model improves from 67.6% to 82.0%. With a 2.14x increase, Llama-3-8B-Instruct achieves an 86.4% accuracy on GSM8K, surpassing GPT-4-Turbo. These results demonstrate that inference-time scaling can be an effective strategy for boosting model performance.

Sample Num.	**1**	**4**	**8**	**16**	**32**
Inference Time	1×	1.26×	1.66×	2.14×	2.61×
Accuracy	67.6	82.0	83.9	86.4	86.2

Table 7: Computational costs of best-of-N sampling of Llama-3-8B-Instruct on GSM8K.

Model	**ApacaEval 2**	**Arena-Hard**	**MixEval**	**MMLU-Redux**	**GSM8K**	**HumanEval**
GPT-4-Turbo	0.03	5.0E-12	6.4E-8	3.2E-6	4.7E-21	1.1E-48
Llama-3-8B-Instruct	2.3E-8	1.5E-14	1.6E-15	0.01	6.8E-26	1.0E-3
Yi-1.5-6B-Chat	0.04	4.2E-7	1.9E-12	1.5E-8	1.1E-34	5.7E-62
Yi-1.5-9B-Chat	1.3E-3	9.7E-12	4.0E-10	0.53	7.5E-62	2.3E-78
Yi-1.5-34B-Chat	0.05	3.4E-3	0.48	8.6E-7	4.7E-42	5.8E-81
Qwen2-7B-Instruct	0.18	8.9E-16	0.57	1.5E-9	3.9E-107	3.8E-82
Mistral-7B-Instruct	1.1E-7	0.66	0.02	0.05	1.6E-78	7.9E-89

Table 8: Significance test for Table 2.