Instance-level Randomization: Toward More Stable LLM Evaluations

Yiyang Li[1,*], Yonghuang Wu[2], Ying Luo[1], Liangtai Sun[1], Zishu Qin[2], Lin Qiu[1], Xuezhi Cao[1], Xunliang Cai[1]

[1]Meituan Group, [2]Fudan University

{liyiyang06,luoying08,sunliangtai,qiulin07,caoxuezhi,caixunliang}@meituan.com

{yonghuangwu21,zsqin23}@m.fudan.edu.cn

Abstract

Evaluations of large language models (LLMs) suffer from instability, where small changes of random factors such as few-shot examples can lead to drastic fluctuations of scores and even model rankings. Moreover, different LLMs can have different preferences for a certain setting of random factors. As a result, using a fixed setting of random factors, which is often adopted as the paradigm of current evaluations, can lead to potential unfair comparisons between LLMs. To mitigate the volatility of evaluations, we first theoretically analyze the sources of variance induced by changes in random factors. Targeting these specific sources, we then propose the instance-level randomization (ILR) method to reduce variance and enhance fairness in model comparisons. Instead of using a fixed setting across the whole benchmark in a single experiment, we randomize all factors that affect evaluation scores for every single instance, run multiple experiments and report the averaged score. Theoretical analyses and empirical results demonstrate that ILR can reduce the variance and unfair comparisons caused by random factors, as well as achieve similar robustness level with less than half computational cost compared with previous methods. Codes and data are available at https://github.com/EricLee8/Instance-level-Randomization.

1 Introduction

With large language models (LLMs) getting stronger (Hurst et al., 2024; DeepSeek-AI et al., 2024; Yang et al., 2025; Meta, 2025), evaluation of them is also becoming harder and more important (Laskar et al., 2024; Chang et al., 2024). One of the biggest challenges of evaluation is that it suffers from instability, where small changes of random factors such as few-shot examples, task descriptions, and even evaluation frameworks can result in significant fluctuations of evaluation scores, and even model rankings (Qian et al., 2024; Xu et al., 2024; Guan et al., 2025; Pan et al., 2025). Figure1 illustrates an example where 7 LLMs are evaluated on Hellaswag (Zellers et al., 2019) dataset. As random factors vary across different runs, model rankings change drastically, with Qwen2.5-7B ranking over as large a range as from 1st to 6th.

Given the above observations, literature can be categorized into the following two groups. One group of works mainly focus on reporting the extreme values or range of fluctuations caused by these random factors. They either utilize Automated Prompt Engineering to find the best or worst prompts, and report the corresponding scores under these prompts (Cao et al., 2024), or design methods to estimate the range of evaluation scores (Madaan et al., 2024; Polo et al., 2024; Alzahrani et al., 2024). Another group of works pay attention on mitigating the instability before the evaluation phase. For example, Kurt et al. 2024 adopt constrained decoding to make outputs more consistent, Irugalbandara 2024 suggest formatting the multiple-choice tasks as cloze tests to avoid the unstable decoding process.

However, few works attempt to enhance the robustness of evaluation (Ngweta et al., 2025). In this paper, we fill the gap of stabilizing evaluations by proposing an embarrassingly simple yet surprisingly effective method to mitigate the bias and variance of evaluations caused by random factors.

In terms of the most adopted paradigm of the current evaluations, it usually adopts a fixed setting of random factors and only runs a single experiment, which imposes biased settings, high variance, and potential unfair comparisons between models due to LLMs' volatility on the random factors. A natural and naive alleviation is to vary the settings of random factors across multiple runs and report the averaged results (Mizrahi et al., 2024). However, this method suffers a high variance and requires a

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3411–3425
November 4-9, 2025

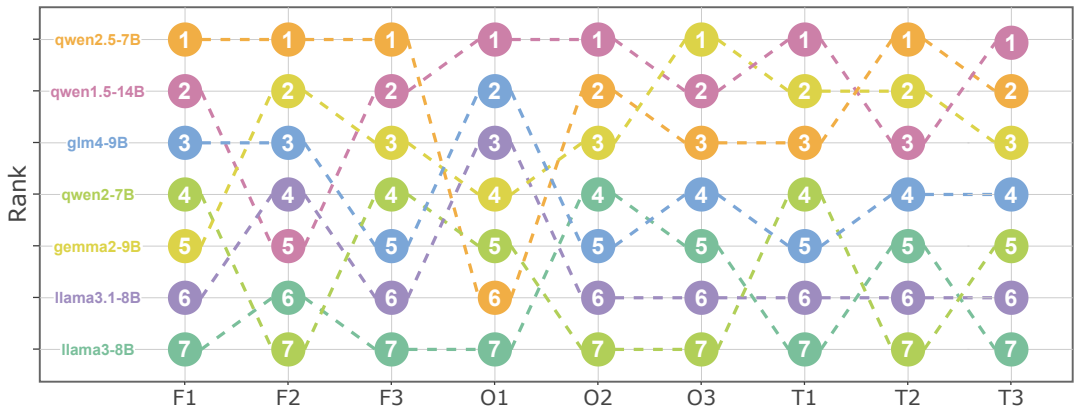

Figure 1: Model ranking variation on Hellaswag with different random factors changing, where "F" represents few-shot examples, "O" represents option labels, and "T" represents task descriptions.

large number of runs to obtain an unbiased and stable result. Based on this, we theoretically analyze the sources of the evaluation instability, based on which we propose our instance-level randomization (ILR) method to stabilize evaluations. Specifically, instead of using a fixed setting for a single experiment, we apply random settings to each of the instances in a benchmark. We show both theoretically and empirically that our proposed ILR is able to reduce the variance faster, using less than 50% computational cost to obtain the same variance.

In this paper, we study few-shot examples, task descriptions, prompt formats, and option labels as the random factors that affect evaluations, whose detailed explanations are in Appendix E. Nevertheless, we underline that without the loss of generality, our ILR can be applied to as many as random factors as we can consider.

To sum up, the contributions of this work are:

- We propose comprehensive theoretical analyses of the sources of the evaluation instability caused by the random factors.
- Based on the theoretical analyses, we propose a simple yet effective instance-level randomization (ILR) method to obtain a more robust evaluation, which can achieve the same robustness level with less than half computational cost compared with previous methods.
- Empirical results solidly validate the theoretical analyses, as well as the effectiveness of our method.

2 Related Work

Instability of evaluations have been discovered and studied by the community for a long time. Studies have shown that random factors such as phrasing and wording of prompts (Errica et al., 2024; Razavi et al., 2025), choice and order of few-shot examples (Xu et al., 2024; Pisarevskaya and Zubiaga, 2025), as well as their format and structure (Ngweta et al., 2025), can lead to large variances in model outputs (Meincke et al., 2025), even affecting factual accuracy (Wang et al., 2024a). Given these factors, studies focus on either reporting or mitigating the evaluation instability.

2.1 Reporting Evaluation Instability

With growing awareness of instability in LLM evaluations caused by random factors, researchers are reporting this phenomenon by quantifying and representing its extent. Studies explore performance under extreme conditions, identifying "best" prompts through Automated Prompt Engineering (Wen et al., 2023; Ye et al., 2024; Kong et al., 2024; Li et al., 2025) and "worst" prompts (Cao et al., 2024). However, focusing on best or worst prompts may not reflect real-world experiences where prompts are diverse and spontaneous, and they require much additional compute, posing challenges for scalable and generalizable evaluation.

Additionally, compared to directly reporting extreme values or variances (Bouthillier et al., 2021; Madaan et al., 2024; Lin et al., 2025), researchers have proposed other statistical metrics, such as performance gaps across different prompts (Sclar et al., 2024; Mizrahi et al., 2024), or median performance and percentile-based metrics (Polo et al., 2024). These methods provide valuable insights into the statistical characteristics of the evaluation instability, but they only focus on the phenomenon itself, rather than offering direct strategies to alleviate its impact on the evaluation process.

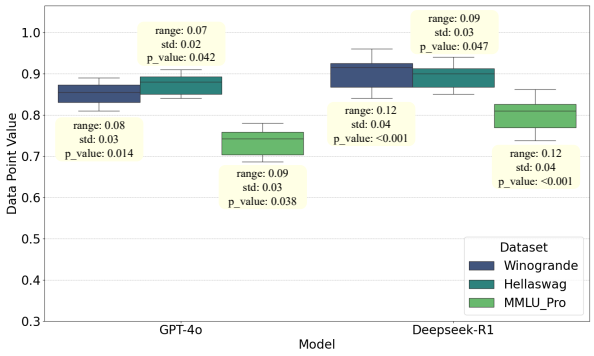

Figure 2: Box plots of different LLMs' scores on the Winogrande, Hellaswag, and MMLU-Pro datasets, where each model is evaluated 8 times under various few-shot example settings. The box plots illustrate the distribution of scores across these 8 runs, where the yellow text box around each box-plot represents the statistical analysis on the results.

2.2 Mitigating Evaluation Instability

Recognizing the presence of instability in LLM evaluations and its potential impact on model rankings, several approaches are proposed to mitigate instability by modifying model outputs. MTP (Irugalbandara, 2024) enhances the reliability and consistency of structured output generation by constrained decoding, but it primarily focuses on stabilizing the generation process rather than the evaluation itself. Similarly, Madaan et al. 2024 propose to reframe multiple-choice tasks as cloze filling to avoid the unstable generation process, thus reducing variance.

Most related to our work, MOF (Ngweta et al., 2025) enhances robustness to prompt format changes by diversifying expression styles in few-shot examples. While effective in reducing performance gaps caused by formatting variations, MOF focuses solely on non-semantic aspects of few-shot examples. In contrast, our approach is applicable to a broader range of random factors, offering a more comprehensive solution to mitigating the evaluation instability.

In summary, existing methods lack the generality to tackle diverse instability sources. Our work fills this gap by proposing a unified framework that stabilizes evaluations against a wide array of random factors.

3 Problems of Current Evaluation

In this section, we show that current evaluation paradigm of LLMs suffers from biased settings, high variance, and potential unfair comparisons between different models. Specifically, there are many random factors that can affect the performance scores of LLMs in a benchmark, such as instructions, task descriptions, prompt formats, and few-shot examples. However, current benchmarks usually adopt a fixed setting of the above random factors, resulting in biased and unstable evaluations of different LLMs.

3.1 Biased Settings

In real-world scenarios, different users may use different settings of the aforementioned random factors to prompt LLMs, which leads to a discrepancy between benchmark scores with fixed factors and the actual user experience.

Formally, the true average score of LLMs on a benchmark should be calculated as follows:

$$S_M = \mathbb{E}_{f\sim\mathcal{F}}\,\mathbb{E}_{(x,y)\sim D}\,\mathbb{I}(M(f(x)) = y), \quad (1)$$

where S_M means the score of model M, f is sampled from $\mathcal{F}$, which is the joint distribution of all random factors that may affect the benchmark performance, (x, y) is the query, where problem x and reference answer y are sampled from the dataset distribution D, and $\mathbb{I}$ is an indicator function that outputs 1 if the output of the LLM matches the reference y.

Current benchmarks only adopt a fixed setting of random factors f_{biased}, resulting in biased estimation of the true average score:

$$S_M^{biased} = \mathbb{E}_{(x,y)\sim D}\,\mathbb{I}(M(f_{biased}(x)) = y). \quad (2)$$

Different choice of f_{biased} can lead to significant changes of benchmark scores, and even shift the model rankings.

3.2 High Variance

Using a fixed setting to evaluate LLMs on a benchmark also introduces high variances of evaluation scores. We show this observation with an example of few-shot evaluations on two strongest models at present: GPT-4o (Hurst et al., 2024) and Deepseek-R1 (DeepSeek-AI et al., 2025). Specifically, we conducted 8 experiments for each model on the Winogrande (Sakaguchi et al., 2021), Hellaswag (Zellers et al., 2019), and MMLU-Pro (Wang et al., 2024b) datasets using different few-shot examples, with each dataset containing 100 instances in Winogrande and Hellaswag, and 500 instances in MMLU-Pro.

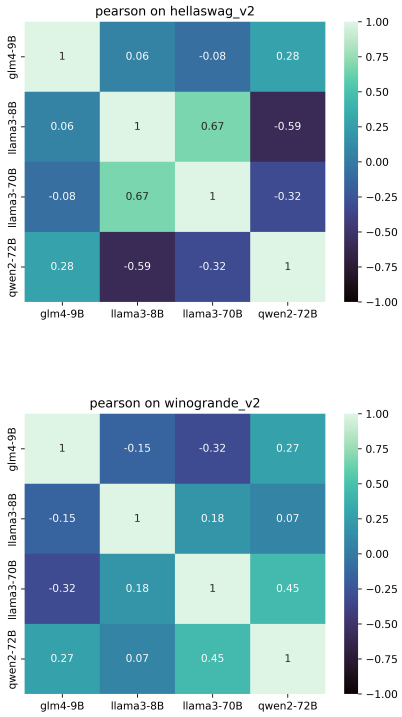

Figure 3: Correlations of different LLMs over different few-shot examples on Hellaswag and Winogrande.

Figure 2 presents box plots of the scores achieved by various LLMs on the three datasets, where the p-value shown in the yellow text box around each box-plot represents the statistical significance of the difference between the best and worst runs out of the 8 experiments, as determined by a paired t-test. As shown in the figure, the choice of few-shot examples has a substantial impact on evaluation scores across all datasets even with the strongest models: the maximum range between the best and worst runs can reach 12%, and in all cases, the differences between the highest and lowest scores are statistically significant ($p < 0.05$).

This observation demonstrates that results obtained from a single experiment with a fixed setting can have high variance, making score comparisons between models unreliable. For more examples of other random factors and on other models, please refer to Appendix C.

3.3 Unfair Comparisons

Another problem of the current fix-setting evaluation is potential unfair comparisons between LLMs. Specifically, different models may exhibit distinct preferences for particular random factors. Taking few-shot evaluation as an example, if we use a fixed set of few-shot examples to evaluate Model A and Model B, we may observe that Model A outperforms Model B. However, if we switch to another set of few-shot examples, the result may be reversed. More generally, this observation manifests as low or even negative correlations between models when multiple experiments are conducted with different random factors.

Figure 3 illustrates the Pearson correlations of different LLMs over different few-shot examples on Hellaswag and Winogrande. We can see from the figure that the correlations are relatively low, where Llama3-8B and Qwen2-72B even show a moderately strong negative correlation. This indicates that using a fixed setting to evaluate models can be potentially unfair if the fixed setting happens to be preferred by one model, and disfavored by another. For correlation analysis of more random factors, please refer to Appendix D.

4 Methodology

In this section, we propose a simple yet effective method using **I**nstance-**L**evel **R**andomization (ILR) over random factors, which can reduce the variance of evaluation scores, and mitigate the instability of model rankings at a lower cost. Figure 4 briefly illustrates our ILR method and the comparison between ILR and the normal evaluation setting. The left side illustrates the normal evaluation paradigm, in which the same set of random factor settings is applied to every instance within a benchmark. Since a given model tends to exhibit consistent preferences toward the same settings, each instance in the benchmark tends to simultaneously overestimate or underestimate the model's true accuracy for that instance. This leads to unfair, biased, and unstable evaluation results. In contrast, the right side presents our proposed ILR method, which employs different settings for all random factors for each instance. This approach offsets preference effects against each other, resulting in fairer, more stable, and less biased evaluation outcomes.

In the subsequent sections, we provide both theoretical and empirical evidence demonstrating that ILR achieves a faster reduction in evaluation variance with fewer runs, thus lowering computational overhead. Besides, we also introduce the Observed Reversal Probability (ORP) as a novel metric to quantify the stability of model rankings. Our experiments confirm that ILR significantly reduces ORP, proving its ability to produce fairer, more stable, and ultimately more reliable evaluation results.

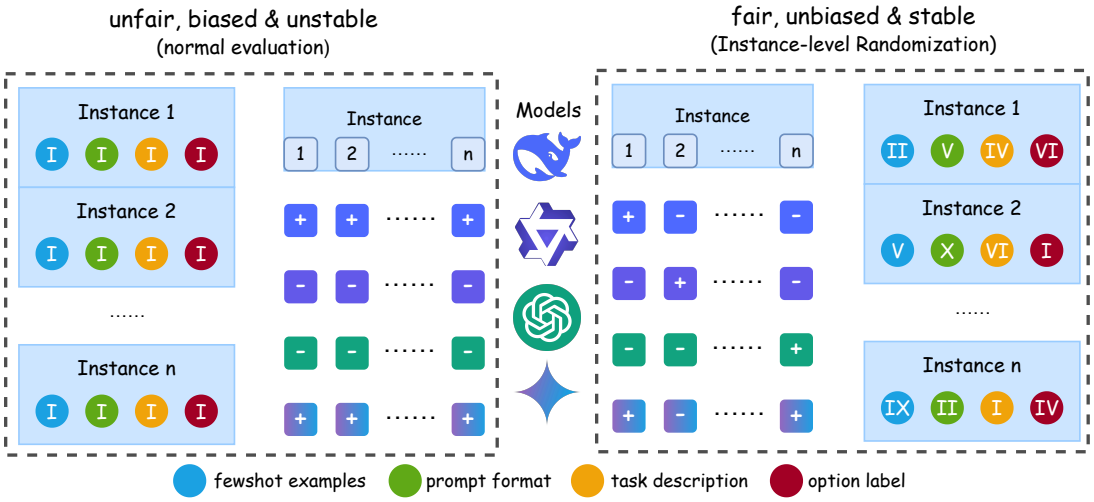

Figure 4: A brief illustration of our ILR method. Different Roman numerals in circles represent different evaluation settings applied to each instance of a benchmark. The plus and minus signs mean that a certain evaluation setting has positive or negative effect on the evaluation result of an instance for a specific model. Different colors of the signs represent different models, who have specific preferences over certain evaluation settings.

4.1 Variance Analysis

Before introducing our ILR method, we analyze the sources of variance when using the mean of n evaluation runs, each considering different random factors, as the benchmark evaluation result. We rewrite Eq. (1) from its expected value form to its sampled mean form. Suppose for the i_{th} evaluation run, the setting of the random factors is f_i, the corresponding accuracy of the benchmark containing m instances $\{(x_k, y_k)\}_{k=1}^{m}$ is marked as $\bar{A}_i = \frac{1}{m}\sum_{k=1}^{m}\mathbb{I}(M(f_i(x_k)) = y_k)$, and the final average accuracy of the benchmark is $\bar{A} = \frac{1}{n}\sum_{i=1}^{n}\bar{A}_i$. For simplicity, we write $\mathbb{I}(M(f_i(x_k)) = y_k)$ as $f_i(x_k)$. So far, we can calculate the variance of the mean of n runs as follows:

$$
\begin{aligned}
Var(\bar{A}) &= Var\left(\frac{1}{n}\sum_{i=1}^{n}\bar{A}_i\right) \\
&= \frac{1}{n^2}\left(\sum_{i=1}^{n}Var(\bar{A}_i) + 2\sum_{1\le i<j\le n}Cov(\bar{A}_i, \bar{A}_j)\right), \\
Var(\bar{A}_i) &= \frac{1}{m^2}\Big(\sum_{k=1}^{m}Var(f_i(x_k)) \\
&\qquad + 2\sum_{1\le k<l\le m}Cov(f_i(x_k), f_i(x_l))\Big).
\end{aligned}
\tag{3}
$$

Combining the equations in Eq. (3), we have:

$$
\begin{aligned}
Var(\bar{A}) = &\frac{1}{n^2}\frac{1}{m^2}\sum_{i=1}^{n}\sum_{k=1}^{m}Var(f_i(x_k)) + \\
&\frac{1}{n^2}\frac{1}{m^2}\sum_{i=1}^{n}\sum_{1\le k<l\le m}2\,Cov(f_i(x_k), f_i(x_l)) + \\
&\frac{1}{n^2}\sum_{1\le i<j\le n}2\,Cov(\bar{A}_i, \bar{A}_j).
\end{aligned}
\tag{4}
$$

Using $\bar{A}_i = \frac{1}{m}\sum_{k=1}^{m}f_i(x_k)$ and the linearity of covariance, $Cov(\bar{A}_i, \bar{A}_j)$ can be expanded as follows:

$$
Cov(\bar{A}_i, \bar{A}_j) = \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l=1}^{m}Cov(f_i(x_k), f_j(x_l)).
\tag{5}
$$

There are three items that consist of $Var(\bar{A})$, each of which is one line in Eq. (4). The first item is the average of the inherent variance of each instance's accuracy, which follows a Bernoulli distribution and cannot be optimized since it only depends on its mean p. The second term represents the pairwise covariance between instances within the dataset, which is typically positive, as instances from the same dataset are usually positively correlated. Reducing the correlation between instances can help minimize this term. The third term is the pairwise covariance of experiment scores across runs with different random factors. Similarly, reducing the correlation between experiments can help minimize this term.

4.2 Instance-level Randomization

To start with, let us revisit the meaning of Eq. (1) and the current evaluation convention first. We sample a setting of random factors f_{biased} from the combinational distribution $\mathcal{F}$, fix it across the sampled data points (x, y) from the data distribution D, and calculate the mean accuracy of these data points as the final score. This convention leads to the aforementioned three problems in Section 3.

To mitigate the problems causing by a fixed f_{biased}, previous works propose to run multiple experiments of different settings and report the average scores (Mizrahi et al., 2024), or report confidence intervals and range distributions (Sclar et al., 2024; Polo et al., 2024). However, they require relatively high number of experiments or more compute to reduce the variance to an acceptable range. Different from previous works, this paper aims to identify methods that can reduce evaluation variance more rapidly with fewer runs.

As mentioned in Section 4.1, the most effective way to reduce variance is to decrease the correlation between instances within the benchmark and the correlation between multiple runs of the benchmark. Therefore, we summarize two points to ensure unbiased and fair evaluation, as well as reduce variance:

- We consider as much random factors as possible when sampling settings, which helps obtain a more unbiased performance estimation.
- Instead of using a fixed setting of random factors across the benchmark, we randomly sample different settings for each instance, which helps decrease the instance-wise correlation thus reducing variance, as well as mitigate potential unfair comparisons caused by different LLMs' preferences over a fixed setting of specific random factors.

Instance-level randomization changes the instance-wise covariance term in Eq. (4) to $Cov(f_i^k(x_k), f_i^l(x_l))$, which means different instances are now evaluated under different random factor settings within an experiment. Also, it changes the inner part of the experiment-wise covariance term in Eq. (5) to $Cov(f_i^k(x_k), f_j^l(x_l))$, which means the differences between experiments now arise not only from a single setting of random factors, but also from distinct random factors applied to each instance. This helps decrease the correlation between instances and also experiments. To explain intuitively, two instances evaluated with different random factors tend to be less correlated than those evaluated with the same random factor.

As for the variance term, the inner part of summation changes from $Var(f_i(x_k))$ to $Var(f_i^k(x_k))$. The change of random factors of an instance may lead to the change of its inherent probability of being answered correctly, thus changing the variance[1]. However, since the change of random factors is randomly sampled, we can assume that the average impact of the variance term to be 0.

All of the above theoretical analyses are empirically validated in Section 5.1.

4.3 Speed of Variance Reduction

In this section, we derive that considering more random factors can accelerate the rate at which the variance of n experiments decreases as n increases, while the ILR method can speed up the rate at which the variance of a single experiment decreases with regard to the number of dataset instances m. Although it is challenging to analyze theoretically how the variance decreases with n when both approaches are combined, our empirical results in Section 5.2 demonstrate that using both methods together can further accelerate the reduction of variance.

We can rewrite the first equation of Eq. (3) as its mean form:

$$\begin{aligned}
\overline{Var} &= \frac{1}{n}\sum_{i=1}^{n} Var(\bar{A}_i), \\
\overline{Cov} &= \frac{2}{n(n-1)}\sum_{1\le i<j\le n} Cov(\bar{A}_i, \bar{A}_j), \\
Var(\bar{A}) &= \frac{1}{n^2}\left(n\overline{Var} + 2\frac{n(n-1)}{2}\overline{Cov}\right) \\
&= \frac{1}{n}\overline{Var} + \frac{n-1}{n}\overline{Cov} \\
&= \frac{1}{n}(\overline{Var} - \overline{Cov}) + \overline{Cov}.
\end{aligned} \tag{6}$$

When considering more random factors, the average variation of single experiment $\overline{Var}$ is unchanged, while the average covariance $\overline{Cov}$ between experiments is reduced. The derivative of $Var(\bar{A})$ with regard to n is $-\frac{1}{n^2}(\overline{Var} - \overline{Cov})$, the absolute value of which will be increased with the decrement of $\overline{Cov}$ since usually we have $\overline{Var} > |\overline{Cov}|$. Therefore, we can achieve lower variance with smaller n. Similarly, we can derive that the speed of single experiment variance reduction with regard to m.

5 Experiments

In this section, we first conduct experiments to empirically validate the theoretical analyses in Section 4.1 and 4.3. After that, we propose a metric called Observational Reversal Probability (ORP) to measure the effectiveness of ILR. The experiments are conducted on Winogrande, Hellaswag, BigBench-Hard (Suzgun et al., 2023), and MMLU-pro (Wang et al., 2024b), with 100 samples on each subset of these datasets if applicable.

[1]The variance of Bernoulli distribution is $p(1-p)$.

	$\text{Corr}^{\text{fixed}}_{\text{instance}}$	$\text{Corr}^{\text{random}}_{\text{instance}}$	$\text{Corr}^{\text{fixed}}_{\text{experiment}}$	$\text{Corr}^{\text{random}}_{\text{experiment}}$	$\text{Var}^{\text{fixed}}_{\text{instance}}$	$\text{Var}^{\text{random}}_{\text{instance}}$
Winogrande	0.177	0.091	0.096	0.073	0.207	0.198
Hellaswag	0.457	0.119	0.213	0.161	0.204	0.197

Table 1: Correlation and variance in Eq. (4) before (marked as *fixed*) and after (marked as *random*) instance-level randomization.

5.1 Covariance Reduction

To validate the theoretical analysis of covariance reduction in Section 4.2, we conduct experiments under fixed settings and ILR settings, and calculate the correlation coefficients at both instance-level and experiment-level, which correspond to the second line of Eq. (4), and Eq. (5), respectively. Specifically, we vary a random factor as the cause of the variance, and fix or randomize others for fixed and ILR settings.

As shown in Table 1, we observe reduction of correlation at both instance-level and experiment-level, where the former drops more significantly after ILR. This is because prior to randomization, instance-level correlations mainly arise from shared random factors; after randomization, these correlations decrease more substantially. In contrast, experiment-level random factors are already different across experiments, even though they are fixed within each experiment, so their correlations are inherently lower and decrease less noticeably after randomization compared to the instance-level. As for the variance term of each instance itself, it remains relatively stable as analyzed in Section 4.2.

5.2 Faster Variance Reduction

To empirically validate the theoretical analysis of faster variance reduction of the proposed method, we conduct experiments by calculating the mean's variance of n experiments with regard to the increase of n. Specifically, we also vary a random factor as the cause of the variance of a single experiment, and adopt three scenarios according to the setting of other random factors in n experiments:

- For each of the n experiments, only one random factor is considered, which is the same setting in previous works (Mizrahi et al., 2024; Ngweta et al., 2025).
- For each of the n experiments, multiple random factors are considered.
- We combine the instance-level randomization and multiple random factors consideration.

We conduct 20 experiments, each of which is repeated 15 times to calculate the standard deviation of a single experiment. For each n, we randomly select n out of the 20 experiments ($n \leq 15$) as the results of n experiments, and repeat this selection process 30 times to get a more accurate estimation. The mean of these selections is used as the standard deviation of the mean across multiple experiments.

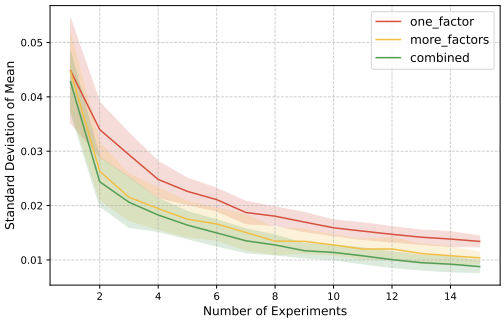

Figure 5: Variation reduction with the number of experiments. The solid lines represent the standard deviation of the mean over n experiments, while the shaded area indicates the estimated standard deviation of this standard deviation, which is calculated from 30 selections.

As shown in Figure 5, the standard deviation drops as the number of experiments increases, which is in accordance with the Law of Large Numbers. Considering more random factors makes the standard deviation drop faster since it reduces the correlation between experiments, as theoretically analyzed in Section 4.3. This observation is also in line with Bouthillier et al. 2021, where they study the effect of random factors on training. Additionally, applying ILR further accelerates the speed of standard deviation reduction. More intuitively, if we aim to reduce the standard deviation to 0.02, previous methods require approximately 6.5 experiments, whereas our ILR only needs about 3. This reduces the evaluation cost by more than $\mathbf{50}\%$.

5.3 Observed Reversal Probability

Besides variance reduction, we show that ILR also helps obtain a more stable and fairer evaluation of model rankings, as well as achieve a more accurate estimation of the models' true performance on a benchmark.

To quantitatively analyze the impact of random factors on model rankings, we introduce the Observed Reversal Probability (ORP). ORP measures the likelihood that the observed ranking of two models (designated A and B) reverses due to the change of random factors.

Let the true accuracies of model A and B be S_A and S_B, respectively. In practice, their observed accuracies are $O_A = S_A + \Delta S_A$ and $O_B = S_B + \Delta S_B$. Here ΔS_A and ΔS_B are normally distributed[2] random variables with standard deviations σ_A and σ_B, and a Pearson correlation coefficient ρ_{AB}. The observed performance difference, $\Delta_S = O_A - O_B$, consequently follows a normal distribution with a mean of $\delta_{AB} = S_A - S_B$ and a variance of $\sigma_A^2 + \sigma_B^2 - 2\rho_{AB}\sigma_A\sigma_B$. ORP measures the probability that the observed ranking is reversed compared with the true ranking due to the change of random factors:

$$\begin{aligned} ORP(\delta_{AB}) &= P(\delta_{AB}\Delta_S < 0) \\ &= \Phi\left(\frac{|\delta_{AB}|}{\sqrt{\sigma_A^2 + \sigma_B^2 - 2\rho_{AB}\sigma_A\sigma_B}}\right). \end{aligned} \tag{7}$$

Here, δ_{AB} denotes the true difference between the two models, while Δ_S represents the observed score difference. A product of them less than zero indicates that the observed ranking is reversed compared to the true ranking. Therefore, the lower ORP, the more stable an evaluation. In practice, σ_A^2, σ_B^2, and ρ_{AB} can be calculated from multiple runs. However, the true difference δ_{AB} is unknown. Therefore, we consider ORP as a function of δ_{AB} and calculate the area under the $ORP - \delta_{AB}$ curve (AUC) as the final ORP value between two models.

5.4 ORP Reduction

To show ILR makes the evaluation more stable, we conduct experiments by varying random factors and calculate the average ORP value between each pair of four models[3] in BBH (Suzgun et al., 2023), Winogrande, and Hellaswag. Figure 6 illustrates the ORP curves before and after ILR, corresponding to either individual random factors or the considering them all. We see from the figure that ILR reduces the ORP for all settings, meaning that the evaluation is less sensitive to the change of random factors after ILR.

We attribute this observation to the following reasons. For one experiment, suppose the accuracy of an instance x_k is composed of three parts:

$$O_M(x_k) = S_M(x_k) + \Delta S_M(x_k, f) + \epsilon_k. \tag{8}$$

Here $O_M(x_k)$ is the observed accuracy of model M on instance x_k, $S_M(x_k)$ is the inherent true accuracy of the model on this instance, $\Delta S_M(x_k, f)$ is the fluctuation term imposed by the random factors f, and ϵ_k is zero-mean random noise. $\Delta S_M(x_k, f)$ is related to both model M and the setting of random factors f, which is also the source of unfair evaluations since different models may have different preferences for certain f. The performance score of a model is calculated as:

$$\begin{aligned} O_M &= \frac{1}{m}\sum_{k=1}^{m}(O_M(x_k, f)) \\ &= \frac{1}{m}\sum_{k=1}^{m}(S_M(x_k) + \Delta S_M(x_k, f) + \epsilon_k) \\ &= S_M + \frac{1}{m}\sum_{k=1}^{m}\Delta S_M(x_k, f) + \frac{1}{m}\sum_{k=1}^{m}\epsilon_k. \end{aligned} \tag{9}$$

Here S_M is the true accuracy of model M on the dataset, $\sum_{k=1}^{m}\epsilon_k$ will gradually converge to 0 as m increases. However, a fixed f will make $\Delta S_M(x_k, f)$ constantly positive or negative, introducing bias of evaluations and unfair comparisons between different LLMs. Fortunately, applying ILR here changes this term to $\sum_{k=1}^{m}\Delta S_M(x_k, f^k)$, where the effects of different f^k can be either positive or negative. As m increases, these effects tend to offset each other, causing this term to approach 0 and avoiding the model bias introduced by fixing f. Therefore, ILR not only enhances the stability of the evaluation, but also renders each experiment less biased and fairer between models.

5.5 ORP as a Meta-Evaluation Metric

In addition to serving as a meta-evaluation metric for algorithms themselves, ORP can also be used as a meta-evaluation metric for datasets. A horizontal comparison of the ORP values in Figure 6 allows us to identify which factors have a greater impact on the dataset, thus guiding the design of algorithms to enhance robustness against them. Vertically, by comparing ORP values across different datasets under the same random factor, we can assess their robustness to random perturbations and select more robust datasets for model evaluation. Overall, ORP is a valuable meta-evaluation metric that offers deeper insights for the evaluation community. We leave further exploration of these analyses as future work.

[2]Values affected by multiple random factors are usually normally distributed, e.g., scores of exams.

[3]Llma3-8B, GLM4-9B, Llama3-70B, and Qwen2-72B.

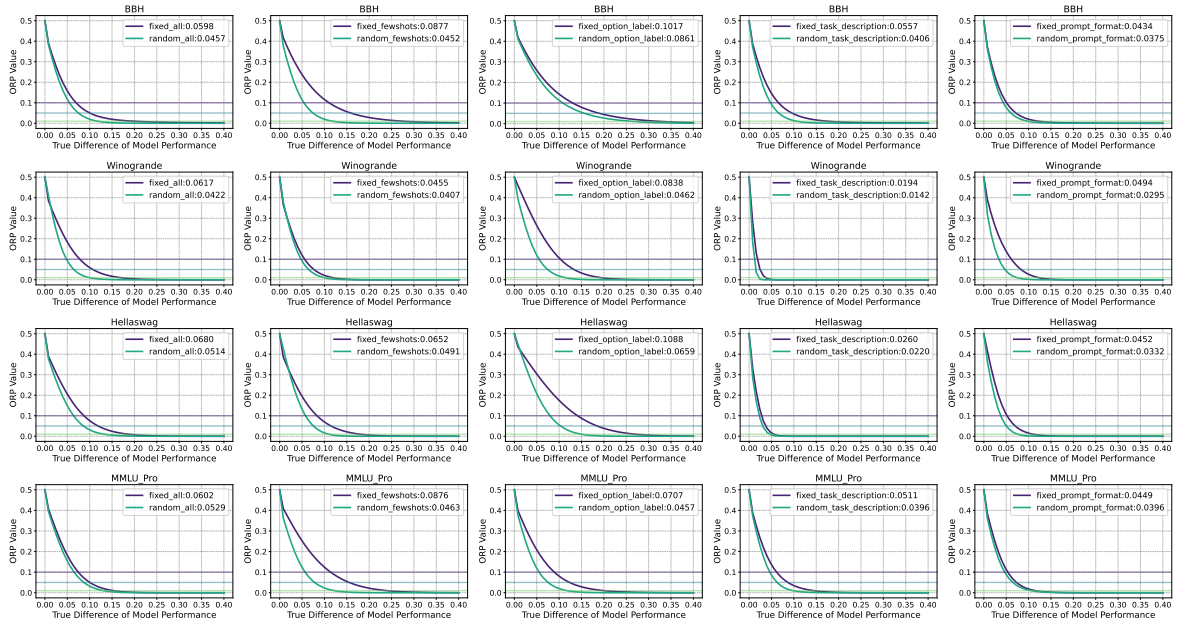

Figure 6: ORP curves between different LLMs on Hellaswag, Winogrande, BBH, and MMLU-Pro. We consider four random factors: few-shot examples, option labels, task descriptions, and prompt formats. Three horizontal lines indicate how much the difference of scores should be to have 90%, 95%, and 99% confidence that one model actually outperforms another, respectively. The values in each legend are the AUCs. The lower the curves (smaller AUCs) the better.

6 Conclusion

This paper tackles the instability of LLM evaluations, where random factors cause score fluctuations and unreliable model rankings. We propose Instance-Level Randomization (ILR), a method that randomizes these factors for each instance and averages results across multiple experiments.

Theoretical analysis and empirical results show ILR effectively reduces variance and unfair comparisons with less than half the computational cost of previous methods. ILR lowers both instance-level and experiment-level correlations and accelerates variance reduction as the number of experiments increases. We also introduce Observed Reversal Probability (ORP) to measure ranking stability, demonstrating that ILR significantly reduces ORP, leading to more dependable evaluations.

To sum up, ILR offers a practical and efficient approach to achieve more robust, fair, and reliable LLM evaluations.

Limitations

While the proposed Instance-Level Randomization (ILR) method demonstrates significant advantages, there are some limitations and avenues for future exploration.

First, we treat the impact of different random factors as independent ones to simplify the empirical validation of the theoretical analysis. However, literature points out that they may correlate with each other (Weber et al., 2023), causing significant difficulties to jointly analyze them. We leave the empirical validation under this circumstance in future work.

Second, the empirical validation in this study primarily focuses on a specific set of common random factors, namely few-shot examples, task descriptions, prompt formats, and option labels. Although ILR is designed to be broadly applicable to as many random factors as can be considered, its performance characteristics when applied to other, potentially more intricate or less common, random factors could be a subject for further investigation.

References

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M. Saiful Bari, and Haidar Khan. 2024. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13787–13805. Association for Computational Linguistics.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. 2021. Accounting for variance in machine learning benchmarks. In *Proceedings of the Fourth Conference on Machine Learning and Systems, MLSys 2021, virtual, April 5-9, 2021*. mlsys.org.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3):39:1–39:45.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 81 others. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Federico Errica, Giuseppe Siracusano, Davide Sanvito, and Roberto Bifulco. 2024. What did I do wrong? quantifying llms' sensitivity and consistency to prompt engineering. *CoRR*, abs/2406.12334.

Bryan Guan, Tanya G. Roosta, Peyman Passban, and Mehdi Rezagholizadeh. 2025. The order effect: Investigating prompt sensitivity in closed-source llms. *CoRR*, abs/2502.04134.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 79 others. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Chandra Irugalbandara. 2024. Meaning typed prompting: A technique for efficient, reliable structured output generation. *CoRR*, abs/2410.18146.

Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Prewrite: Prompt rewriting with reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024 - Short Papers, Bangkok, Thailand, August 11-16, 2024*, pages 594–601. Association for Computational Linguistics.

Will Kurt, Remi Louf, and Cl é mentine Fourrier. 2024. Improving prompt consistency with structured generations.

Md. Tahmid Rahman Laskar, Sawsan Alqahtani, M. Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee-Wei Tan, Md. Rizwan Parvez, Enamul Hoque, Shafiq Joty, and Jimmy Huang. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 13785–13816. Association for Computational Linguistics.

Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *CoRR*, abs/2502.11560.

Feng Lin, Dong Jae Kim, Zhenhao Li, Jinqiu Yang, and Tse-Hsun (Peter) Chen. 2025. Robunfr: Evaluating the robustness of large language models on non-functional requirements aware code generation. *CoRR*, abs/2503.22851.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. 2024. Quantifying variance in evaluation benchmarks. *CoRR*, abs/2406.10229.

Lennart Meincke, Ethan Mollick, Lilach Mollick, and Dan Shapiro. 2025. Prompting science report 1: Prompt engineering is complicated and contingent. *CoRR*, abs/2503.04818.

AI Meta. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. *https://ai. meta. com/blog/llama-4-multimodal-intelligence/, checked on*, 4(7):2025.

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. *Trans. Assoc. Comput. Linguistics*, 12:933–949.

Lilian Ngweta, Kiran Kate, Jason Tsay, and Yara Rizk. 2025. Towards LLMs robustness to changes in prompt format styles. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 529–537, Albuquerque, USA. Association for Computational Linguistics.

Jane Pan, Ryan Shar, Jacob Pfau, Ameet Talwalkar, He He, and Valerie Chen. 2025. When benchmarks talk: Re-evaluating code llms with interactive feedback. *CoRR*, abs/2502.18413.

Dina Pisarevskaya and Arkaitz Zubiaga. 2025. Zero-shot and few-shot learning with instruction-following llms for claim matching in automated fact-checking. *arXiv preprint arXiv:2501.10860*.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. 2024. Efficient multi-prompt evaluation of llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. 2024. VarBench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16131–16161, Miami, Florida, USA. Association for Computational Linguistics.

Amir Hossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. 2025. Benchmarking prompt sensitivity in large language models. In *Advances in Information Retrieval - 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6-10, 2025, Proceedings, Part III*, volume 15574 of *Lecture Notes in Computer Science*, pages 303–313. Springer.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

Hongchen Wang, Kangming Li, Scott Ramsay, Yao Fehlis, Edward Kim, and Jason Hattrick-Simpers. 2024a. Evaluating the performance and robustness of llms in materials science q&a and property predictions. *CoRR*, abs/2409.14572.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Lucas Weber, Elia Bruni, and Dieuwke Hupkes. 2023. Mind the instructions: a holistic evaluation of consistency and interactions in prompt-based learning. In *Proceedings of the 27th Conference on Computational Natural Language Learning, CoNLL 2023, Singapore, December 6-7, 2023*, pages 294–313. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Derek Xu, Tong Xie, Botao Xia, Haoyu Li, Yunsheng Bai, Yizhou Sun, and Wei Wang. 2024. Does few-shot learning help LLM performance in code synthesis? *CoRR*, abs/2412.02906.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao

Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. 2024. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 355–385. Association for Computational Linguistics.

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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

A Computational Experiments

We conduct experiments on Qwen2-72B, Llama3-70B, GLM4-9B, Llama3-8B, Gemma2-9B, Qwen2.5-7B, Qwen1.5-14B. For models with less than 14B parameters, we deploy them on 1 single A100-80GB GPU. It takes about 1 hour to run $10,000$ instances with vLLM framework. For models more than 14B parameters, we deploy them on 4 A100-80GB GPUs. It takes about 2 hours to run $10,000$ instances with vLLM framework.

B Information About Use Of AI Assistants

We use GPT-4o to polish some sentences of our paper.

C Box plots of different LLM scores

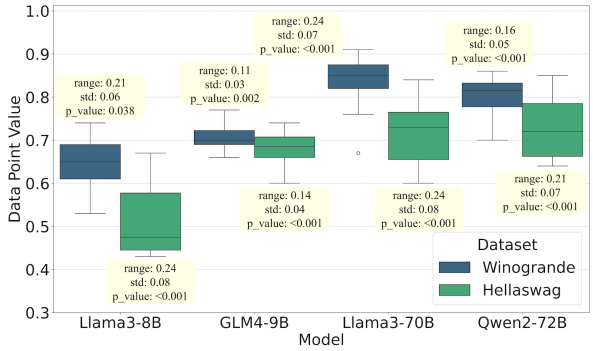

Figure 7: Box plots of different LLMs' scores on the Winogrande and Hellaswag datasets, where each model is evaluated 8 times under different option label settings.

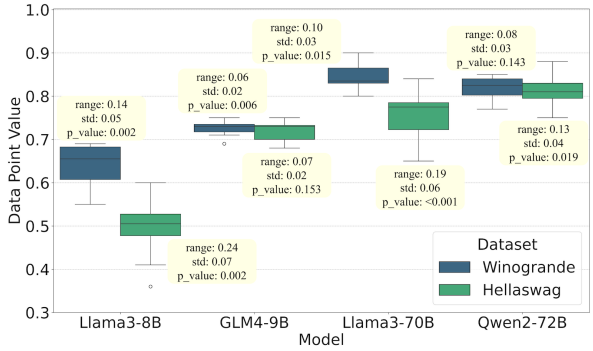

Figure 8: Box plots of different LLMs' scores on the Winogrande and Hellaswag datasets, where each model is evaluated 8 times under different task description settings.

D Correlations of different LLMs

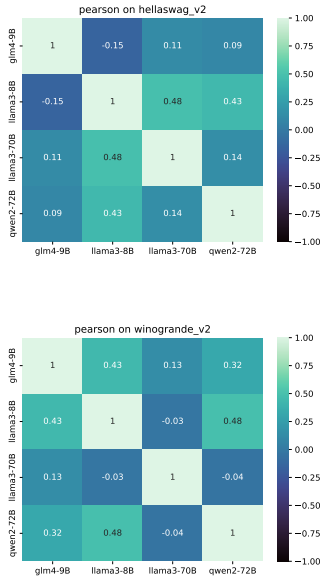

Figure 9: Correlations of different LLMs over different option labels on Hellaswag and Winogrande.

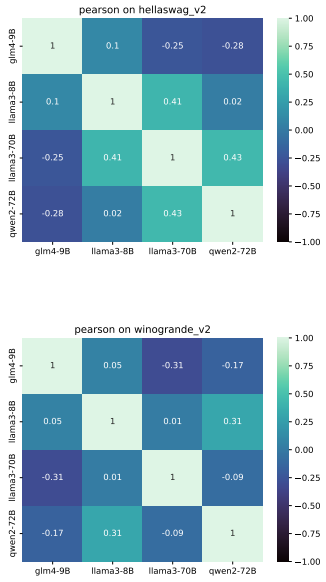

Figure 10: Correlations of different LLMs over different task descriptions on Hellaswag and Winogrande.

E Example of Different Random Factors

We study few-shot examples, task descriptions, prompt formats, and option labels as the random factors that affect evaluations. As shown in Figure 11, few-shot examples and option labels are illustrated in red and green, respectively. We define the combination of question prefixes, option prefixes, and answer prefixes, as the prompt format, which is shown in blue. The combination of introductions of the task and the Chain-of-Thought (Wei et al., 2022) prompts marked in orange is defined as task descriptions.

Example of raw data

Given a context and multiple options, choose the most reasonable continuation.
Question:
[header] How to take care of a budgie . . .
Options:
A. Cockatiels only need a cage . . .
B. [substeps] They should also . . .
C. Cockatiels will be more interested . . .
D. The cage should be at least . . .
Let us do this task step by step.
The solution is: The question asks . . .
[OTHER FEW-SHOT EXAMPLES]
Question:
Several replays of acrobatic moves . . .
Options:
A. go across the screen directing . . .
B. are written in yellow letters . . .
C. are shown followed on the . . .
D. are then repeated on the . . .
Let us do this task step by step.
The solution is:

Figure 11: The example of raw data is shown above, where the red texts are few-shot examples, the blue texts are prompt formats, the green texts are option labels, and the orange texts are task descriptions.

Example of different few-shot examples

Given a context and multiple options, choose the most reasonable continuation.
Question:
A soccer player is held back from . . .
Options:
A. are curling on the sidelines. . . .
B. kick the ball to one another, . . .
C. play soccer in an outdoor field . . .
D. huddle up holding a goal . . .
Let us do this task step by step.
The solution is: To solve this problem . . .
[OTHER FEW-SHOT EXAMPLES]
Question:
Several replays of acrobatic moves . . .
Options:
A. go across the screen directing . . .
B. are written in yellow letters . . .
C. are shown followed on the . . .
D. are then repeated on the . . .
Let us do this task step by step.
The solution is:

Figure 12: The example of different few-shot examples.

Example of different prompt formats

Given a context and multiple options, choose the most reasonable continuation.
Here is a question:
[header] How to take care of a budgie . . .
Options:
A. Cockatiels only need a cage . . .
B. [substeps] They should also . . .
C. Cockatiels will be more interested . . .
D. The cage should be at least . . .
Let us do this task step by step.
The solution is: The question asks . . .
[OTHER FEW-SHOT EXAMPLES]
Here is a question:
Several replays of acrobatic moves . . .
Here are the options:
A. go across the screen directing . . .
B. are written in yellow letters . . .
C. are shown followed on the . . .
D. are then repeated on the . . .
Let us do this task step by step.
Here is the solution:

Figure 13: The example of different prompt formats.

Example of different option labels

Given a context and multiple options, choose the most reasonable continuation.
Question:
[header] How to take care of a budgie . . .
Options:
(1) Cockatiels only need a cage . . .
(2) [substeps] They should also . . .
(3) Cockatiels will be more interested . . .
(4) The cage should be at least . . .
Let us do this task step by step.
The solution is: The question asks . . .
[OTHER FEW-SHOT EXAMPLES]
Question:
Several replays of acrobatic moves . . .
Options:
(1) go across the screen directing . . .
(2) are written in yellow letters . . .
(3) are shown followed on the . . .
(4) are then repeated on the . . .
Let us do this task step by step.
The solution is:

Figure 14: The example of different option labels.

Example of different task descriptions

Given a context and several options, select the most logical continuation.
Question:
[header] How to take care of a budgie . . .
Options:
A. Cockatiels only need a cage . . .
B. [substeps] They should also . . .
C. Cockatiels will be more interested . . .
D. The cage should be at least . . .
We will address this task gradually.
The solution is: The question asks . . .
[OTHER FEW-SHOT EXAMPLES]
Question:
Several replays of acrobatic moves . . .
Options:
A. go across the screen directing . . .
B. are written in yellow letters . . .
C. are shown followed on the . . .
D. are then repeated on the . . .
We will address this task gradually.
The solution is:

Figure 15: The example of different task descriptions.