SeaPO: Strategic Error Amplification for Robust Preference Optimization of Large Language Models

Jun Rao[1], **Yunjie Liao**[1], **Xuebo Liu**[1*], **Zepeng Lin**[1],
Lian Lian[2], **Dong Jin**[2], **Shengjun Cheng**[2], **Jun Yu**[3], **and Min Zhang**[1]
[1]Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen
[2]Huawei Cloud Computing Technologies Co., Ltd.
[3]School of Intelligence Science and Engineering, Harbin Institute of Technology, Shenzhen
{rao7jun,yunjie445,zepenglin11}@gmail.com
{liuxuebo,yujun,zhangmin2021}@hit.edu.cn
{lianlian3,jindong2,chengshengjun}@huawei.com

Abstract

Existing alignment methods for preference optimization of large language models (LLMs) aim to enhance model performance by utilizing pairs of positive and negative samples. However, due to the limited capacity of models in scoring or generating responses, the quality of positive and negative samples may become similar during training, which complicates optimization for preference learning. To address this issue, we introduce SeaPO, a **S**trategic **E**rror **A**mplification method that leverages three error types commonly occurring in LLMs to introduce specific error patterns into the model **P**reference **O**ptimization. This strategy ensures that negative samples are more erroneous than positive samples and preference-based training is employed to mitigate the occurrence of these errors, thereby enhancing model performance. Evaluations across five capability dimensions and different model scales (1.5B to 14B) demonstrate that the generated data significantly improved overall model performance, particularly in terms of truthfulness, with improvements of 5–10 percentage points observed. Further analysis reveals that task performance varies depending on the error types introduced. Injecting the most common error types improves performance in related tasks, while a mix of error types leads to a broader performance enhancement: most tasks show stable improvements, while a few tasks exhibit significant gains.

1 Introduction

Since the advent of GPT4 (OpenAI, 2023), data have become the major affect factor in large language models (LLMs) of capability (Wang et al., 2023; Qi et al., 2023; Deng et al., 2024). Extensive research has explored various aspects of data, including knowledge distillation (Peng et al., 2023; Rao et al., 2023a,b; Bao et al., 2023; Yang et al., 2024a), generation (Zhang et al., 2025; Yang et al., 2025), filtering (Jiang et al., 2023; Rao et al., 2025), selection (Chen et al., 2024a; Xiao et al., 2024), and training (Rao et al., 2024). Past work primarily focused on optimizing samples in the supervised fine-tuning phase, i.e., improving the quality (Longpre et al., 2023; Wei et al., 2022; Xie et al., 2023; Chen et al., 2024a; Touvron et al., 2023; Dubey et al., 2024), diversity (Xu et al., 2023) and length (Zhao et al., 2024a) of the training samples. Recent work has also explored the effect of data in DPO (Rafailov et al., 2023), using methods such as rejection sampling (Yuan et al., 2023; Cui et al., 2024), iteration of negative samples (Chen et al., 2024b; Dong et al., 2025) (which progressively improves the quality of negative samples), and applying constraints to prevent overfitting the model on existing data (Zhao et al., 2024b), including length constraints (Meng et al., 2024; Han et al., 2024) and regularization constraints (Pal et al., 2024).

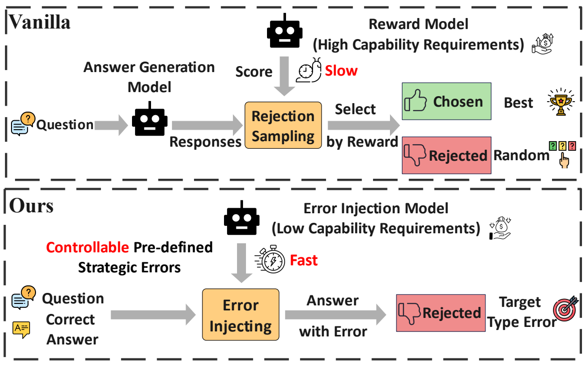

Figure 1: Due to the limitations of the reward model and the sampling, the rejection may be as good as the chosen one, making preference-based optimization more difficult. In the vanilla approach, positive and negative samples are obtained through multiple sampling and a scoring model, which are required to identify positive and negative samples. In contrast, our method only requires introducing specific error types to the original responses without the need for rejection sampling.

[*] Corresponding Author

However, due to the inherent limitations of a model's capabilities (Jiang et al., 2023; Bai et al., 2023; Zhu et al., 2024), achieving human-like precision in specific tasks (Zhao et al., 2024c), such as evaluating sentence quality or generating accurate responses (Wei et al., 2025; Han et al., 2025) to complex questions, remains challenging. This results in scoring models potentially misjudging (Zhao et al., 2024d, 2025) positive and negative samples, leading to minimal differences between them and, consequently, difficulties in preference-based training, which can result in potential logical or hallucination generation.

To address this challenge, we propose SeaPO, which utilizes error-injected negative samples that have been largely overlooked in prior studies. Specifically, we define three strategic errors commonly occurring in LLMs and introduce a straightforward error-injected negative sample generation method to produce these target error types. After that, SeaPO employs an error-focused preference optimization objective, aiming to reduce the probability of negative samples and, in turn, minimize the likelihood of the introduced strategic errors, thereby enhancing overall model performance. SeaPO reduces the dependence on powerful models, requiring only minimal error injection ability and minimizes computational overhead by eliminating the scoring process, as shown in Figure 1.

Through extensive experimental validation, we demonstrate that SeaPO effectively generates more erroneous samples compared to the original negative samples, thereby enhancing both overall and specific model capabilities, such as mathematical reasoning, code solving, and truthfulness, through negative sample augmentation with these strategic errors. The results show improvements of up to 18.8% in Llama3-8B-Instruct and 12.3% in Qwen2.5-7B-Instruct on truthfulness, surpassing existing methods such as Cui et al. (2024). Additionally, ablation experiments highlight the significance of both error definition and error injection.

Our main contributions are as follows:

- We introduce SeaPO, which innovatively utilizes self-rewritten negative samples with preference optimiztation, effectively reducing the occurrence of undesirable error modes (§3).

- SeaPO demonstrates exceptional versatility by achieving strong performance across a wide range of datasets and model scales (1.5B to 14B), proving its robustness and scalability in diverse settings (§4).

- Our exploration of error definitions and the error injection reveals the impact of negative sample diversity, quality, and the severity of their errors on the final performance (§5).

2 Related Work

2.1 Preference Learning Data

Many studies (Guo et al., 2024; Li et al., 2024; Chen et al., 2024b; Kim et al., 2024a) have recognized the value of self-generated data for model enhancement, typically using iterative improvement (Liang et al., 2024; Chen et al., 2024b; Dong et al., 2025) or self-rewriting (Li et al., 2024) to refine training sample quality and boost model performance. UltraFeedback (Cui et al., 2024) constructs preference datasets via multi-model rejection sampling, selecting the highest-scoring response as positive and randomly rejecting others as negative. While existing methods focus on generating more correct answers, they struggle to improve the quality of erroneous samples (Huang et al., 2024; Li et al., 2024). Notably, Huang et al. (2023) leverages the HHH Criteria to construct credibility-focused incorrect answers, but this approach is limited to TruthfulQA and not validated for reasoning or mathematical tasks. Contemporaneous work (Xu et al., 2024) randomly introduced errors to enhance the results of a single task like math. SeaPO introduces specific types of errors, improving the performance of tasks prone to such errors.

2.2 Preference Learning Algorithm

Preference learning trains models to align outputs with human preferences or goals, often using techniques like reward shaping or reinforcement learning to optimize performance and minimize unexpected behaviors. Rafailov et al. (2023) propose Direct Preference Optimization (DPO) to efficiently train large models for knowledge alignment using preference rankings instead of reward models. DPO optimizes classification loss from preference data, making implementing it simpler than reinforcement learning from human feedback. Recent improvements include: Smaug (Pal et al., 2024) uses the addition of regularity to improve the effectiveness of DPO. ORPO (Hong et al., 2024) strongly rewards the choice of answers by modifying the optimisation objective as well as removing the reference model. SimPO (Meng et al., 2024)

Strategic Errors	Descriptions	Tasks where This Error Commonly Occurs
Correctness	Mistakes related to factual accuracy or calculations, such as incorrect facts, reasoning errors, translation issues, or improper use of tools and formulas.	Mathematical Calculation, Fact Verification, Text Summarization
Logic	Issues where the reasoning or argumentation is flawed, such as contradictions, unsupported conclusions, circular reasoning, or failure to follow a logical sequence in problem-solving or explanation.	Mathematical Reasoning, Commonsense Reasoning, Algorithm Design
Hallucination	Instances where the model generates information that is completely fabricated or false, without basis in reality or relevant data, often resulting in the creation of nonexistent facts or misleading details.	Knowledge-based QA, Code Generation, Machine Translation

Table 1: Strategic errors and their corresponding task types.

reconstructs the training objective by adopting the length regularity reward. KTO (Ethayarajh et al., 2024) utilizes prospect theory to optimize model alignment with human feedback by directly maximizing a utility function based on binary signals, minimizing the need for preference data. These algorithms sequentially increase the probability of positive samples and decrease the probability of negative samples. We focus on the data, leveraging it to optimize the final outcome by applying the same optimization method across varying datasets.

3 Strategic Error Amplification

3.1 Overview

In Figure 2, SeaPO consists of two components: Error-Injected Negative Sample Generation and Error-Focused Preference Optimization. 1) Error Generation: We predefine common LLM error types (correctness, logic, hallucination, termed Strategic Errors) and use the target model itself to post-edit training responses, intentionally injecting errors in specified proportions to address targeted capacity gaps. 2) Preference Optimization: Using KTO, we optimize both original positive samples and error-injected negative samples to refine the model's ability to distinguish correct/incorrect responses, enhancing accurate answer generation.

3.2 Definition of Strategic Errors

Model development often involves analyzing problematic cases in specific scenarios to identify the types of issues that need to be addressed. It is assumed that this process has been analyzed manually. We define strategic errors as inaccuracies, logical flaws, or misleading outputs that frequently occur in the responses generated by LLMs. These errors are systematically identified based on common mistakes frequently observed in model outputs, enabling a more nuanced evaluation of their performance and robustness. Specifically, we categorize these strategic errors into three primary types: correctness errors, logic errors, and hallucinations. Table 1 provides a detailed classification of these error types and highlights the tasks most susceptible to each category, offering a comprehensive overview of strategic errors and their relevance across different tasks. By systematically identifying and analyzing these strategic errors, we aim to deepen our understanding of the limitations and capabilities of LLMs across various task domains.

3.3 Error-Injected Negative Sample Generation

The goal of SeaPO is to suppress the model's tendency to commit specific types of errors when answering questions in task-specific scenarios. To achieve this, constructing error-injected samples that closely resemble real-world mistakes becomes crucial. Negative samples that exhibit a moderate level of error, neither excessively incorrect nor overly accurate, can enhance the final outcomes of preference training. We generate negative samples by leveraging the training set and predefined error types. Specifically, we prompt the LLM to modify the original correct answers by injecting errors corresponding to each identified error category, with the prompt we used detailed in Appendix A.1. This process can be formalized as follows: Let $\mathcal{E} = \{e_c, e_l, e_h\}$ denote the set of predefined error types. For a given question x and its correct answer y, we define a specific error instance $e \in \mathcal{E}$. The error-injected answer (negative sample) is then generated using the following function:

$$a_{\text{error}} = \text{Injector}(x, y, e), \quad (1)$$

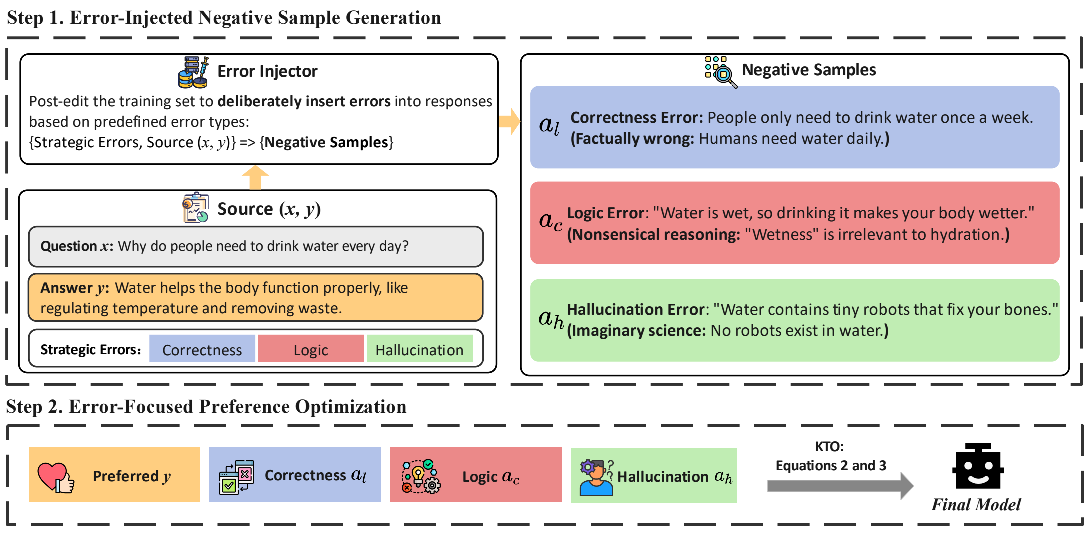
Step 1. Error-Injected Negative Sample Generation
Error Injector
Post-edit the training set to deliberately insert errors into responses based on predefined error types:
{Strategic Errors, Source (x, y)} => {Negative Samples}
Source (x, y)
Question x: Why do people need to drink water every day?
Answer y: Water helps the body function properly, like regulating temperature and removing waste.
Strategic Errors: Correctness Logic Hallucination
Negative Samples
a_l Correctness Error: People only need to drink water once a week. (Factually wrong: Humans need water daily.)
a_c Logic Error: "Water is wet, so drinking it makes your body wetter." (Nonsensical reasoning: "Wetness" is irrelevant to hydration.)
a_h Hallucination Error: "Water contains tiny robots that fix your bones." (Imaginary science: No robots exist in water.)
Step 2. Error-Focused Preference Optimization
Preferred y
Correctness a_l
Logic a_c
Hallucination a_h
KTO: Equations 2 and 3
Final Model

Figure 2: An overview of the SeaPO. The entire process consists of two parts: 1) Error-Injected Negative Sample Generation. 2) Error-Focused Preference Optimization.

where a_{error} represents the answer with the injected error, and e specifies the concrete error instance to be introduced into the question x. The error injection process is performed by the model itself, guided by the injector prompt. We utilize three distinct error types alongside the original question-and-answer pair to systematically inject errors. In this setup, the model generates responses tailored to each injected error type for every question type.

While our method is straightforward, the specific error categories are rooted in human empirical analysis. Currently, the construction of our prompts is generated by GPT-4o (see the specific prompt for generating the prompting words in Appendix A.1). We also provide examples of the error-injected samples for each of the three error types in Appendix A.2, which demonstrate the variations in the injected errors across different question types. These three types of responses address the original question from different perspectives, yet each intentionally incorporates the respective error type. This process generates diverse negative samples simulating realistic errors, enhancing model robustness in preference optimization.

3.4 Avoiding Pitfalls: Error-Focused Preference Optimization

KTO (Ethayarajh et al., 2024) is an effective algorithm that reduces the impact of negative samples in real-world scenarios. Although acquiring prompt-chosen-reject paired data remains challenging, it is still possible to evaluate output acceptability. Importantly, KTO's design eliminates the requirement for matched data pairs during training, making it particularly well-suited for scenarios focused on error cases. At its core, KTO is grounded in the value function, which models human decision-making under uncertainty. The value function captures the phenomenon of loss aversion, where humans are more sensitive to losses than equivalent gains. The value of an outcome z is calculated relative to a reference point z_0, with the function exhibiting concavity for gains and convexity for losses:

$$v(z;\lambda,\alpha,z_0)=\begin{cases}(z-z_0)^\alpha & \text{if } z\geq z_0\\ -\lambda(z_0-z)^\alpha & \text{if } z<z_0\end{cases}, \quad (2)$$

where α controls risk attitudes, and λ represents loss aversion. The core objective of KTO is to optimize the expected utility of model generations:

$$L_{KTO}(\pi_\theta,\pi_{\text{ref}})=\mathbb{E}x,y\sim D[\lambda_y-v(r_\theta(x,y))], \quad (3)$$

where $r_\theta(x,y)=\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ is the reward for output y given input x, and $v(r_\theta(x,y))$ is the corresponding value based on the reward. The reference model π_{ref} serves as a baseline distribution (instruct model), guiding the model's outputs while maintaining their alignment with human expectations.

3.5 Complexity Discussion

Typical negative sample construction uses rejection sampling (Yuan et al., 2023), generating multiple responses per problem without explicit output pattern control, risking similar-quality outputs. For example, UltraFeedback (Cui et al., 2024) randomly

Model	Method	Math		Reasoning	Coding	Knowledge	Truthful		Avg.
		MATH	GSM	BBH	HumanEval	MMLU	MC1	MC2	
Llama3-8B	Instruct	28.0	75.1	**68.5**	75.5	**65.7**	36.2	51.8	57.3
	+Vanilla	**28.1**	**75.3**	67.6	77.5	65.3	38.9	56.3	58.4
	+SeaPO	25.8	72.2	62.9	**79.5**	64.7	**55.0**	**66.1**	**60.9**
Qwen2.5-1.5B	Instruct	30.4	61.0	**45.6**	**70.5**	**59.5**	30.5	46.0	49.1
	+Vanilla	27.8	56.4	43.4	**70.5**	59.2	31.5	48.0	48.1
	+SeaPO	**30.9**	**64.1**	44.7	69.2	59.3	**39.3**	**55.0**	**51.8**
Qwen2.5-7B	Instruct	47.2	72.5	**67.5**	88.7	72.8	42.4	59.3	64.3
	+Vanilla	46.8	73.4	59.4	**90.9**	**72.9**	51.0	65.3	65.7
	+SeaPO	**56.5**	**75.4**	67.0	88.6	72.5	**60.2**	**70.9**	**70.2**
Qwen2.5-14B	Instruct	50.1	85.6	73.7	89.1	**79.7**	51.5	69.2	71.3
	+Vanilla	50.1	86.8	73.1	**89.2**	**79.7**	52.0	**69.6**	71.5
	+SeaPO	**53.9**	**90.7**	**78.9**	88.5	79.3	**52.5**	69.1	**73.3**

Table 2: Main results on multiple test sets for multiple models. The results show that our SeaPO can deliver superior results compared to the existing method over multiple model series.

constructs negatives via multi-model inference (Alpaca-GPT-4 (Peng et al., 2023), Llama (Dubey et al., 2024), Vicuna (Zheng et al., 2023), $\geq$7B params), selecting three negatives and one positive per problem with a GPT4 scoring model, then randomly pairing one negative-positive pair for training. In contrast, our method eliminates the need for an auxiliary scoring model, drastically reducing data construction overhead. This optimization enhances both construction and generation efficiency.

4 Experiments

4.1 Setup

Baseline and Models Following previous work (Meng et al., 2024; Rafailov et al., 2023) of preference optimization, we use UltraFeedback (Cui et al., 2024) including 64K prompts and 256K responses (a prompt generates four responses, one of which is used as a positive sample and one of the remaining three is randomly selected as a negative sample), as the initial dataset for preference learning with KTO as a baseline. We use the Qwen2.5 series (1.5B, 7B and 14B) and Llama3-8B-Instruct as the instruction models for our error-injecting before preference learning. We use the Qwen2.5-7B-Instruct as the base model for the analysis experiments.

Evaluations Math, reasoning, coding, knowledge, and truthfulness are core LLM capabilities, evaluated via out-domain benchmarks. In the evaluations, we examine the model's performance in these areas, employing out-domain benchmarks to assess its ability to assimilate math (MATH (Hendrycks et al., 2021b) and GSM (Cobbe et al., 2021)), executing complex reasoning (BBH (Suzgun et al., 2023)), coding (HumanEval (Chen et al., 2021)), knowledge (MMLU (Hendrycks et al., 2021a)) and truthful (Lin et al., 2022) tasks. We use the exact match as the evaluation metric to assess the models' ability of Math and Reasoning to provide correct answers. We use the HumanEval to evaluate the models' capability to generate functionally correct programs from docstrings, with pass@10 as the evaluation metric. We employ the MMLU to measure factual knowledge, using accuracy as the evaluation metric. For Truthful, since there are multiple correct answers, apart from standard accuracy (MC1), we also report the ratio of likelihood sum of correct answers overall candidate answers (MC2).

Training Details The model was trained using a learning rate of 5×10^{-6}, following a cosine decay strategy for the learning rate schedule. A per-device batch size of 2 was used during training, and to achieve an effective total batch size of 64, we employed gradient accumulation over 8 steps, which optimized memory usage during training. The training process utilized a multi-GPU distributed setup with 4 GPUs and was initialized with a random seed of 42 for reproducibility. For optimization, we used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. Training proceeded for a total of 5731 steps, determined by the total number of training samples and the batch size. Regarding the hyperparameters of KTO, we set $\beta = 0.1$. Additionally, we fine-tuned the model in all experiments using LoRA (Hu et al., 2022). The LoRA configuration included a rank of 8, and a scaling factor α of 16, without dropout.

4.2 Main Results

Table 2 demonstrates that constructing diverse negative samples for preference optimization is effective, as reducing the occurrence of error patterns leads to improved model performance. Compared to the original instruct model and UltraFeedback's KTO training, our SeaPO consistently outperforms across different models, with notable improvements on error-rich datasets such as MATH and TruthfulQA. For instance, Qwen2.5-7B achieves gains of +10 points on MATH and +12 points on TruthfulQA. These improvements are consistent across various architectures (e.g., Llama3-8B, with a +20 point increase) and model scales. Smaller models ($<$8B) benefit the most from error-type injection to reduce hallucinations, while larger models ($\geq$8B) show enhanced reasoning capabilities through the reduction of incorrect patterns.

Notable declines in some models (e.g., Qwen2.5-1.5B 3 points in Math) may stem from low-quality injected errors or limited capabilities, as further analyzed in Tables 6 and 7. Minimal changes occur in strong areas like coding (scores 90) and knowledge-based MMLU. Knowledge-related capabilities are primarily determined during the pre-training phase, with post-training having only a limited impact, typically resulting in changes of less than two points (Wang et al., 2023; Yang et al., 2024b). For code-related tasks, baseline performance is already high, and the evaluation set contains only 164 samples, making the scores highly sensitive to individual errors. Consequently, variations within two points are better interpreted as random fluctuations rather than meaningful changes. In contrast, other tasks, such as GSM, include larger evaluation sets (e.g., 1,319 samples), where noticeable score improvements (e.g., +10 points) require consistently correct responses across many instances. These differences in dataset size and difficulty explain the observed score variability in code and knowledge.

5 Analysis

5.1 Analysis on Strategic Error Definition

Setup Table 3 presents a comparison of the effects of different error types across multiple tasks. The evaluation includes baselines: ID 1 (None), representing the original instruction-tuned model, and ID 2 (Untargeted), which corresponds to results obtained by rewriting positive samples without type-specific constraints. We report results for three error types (IDs 3–5) and their mixed variants (IDs 6–10). IDs 6–8 and ID 10: These involve mixing responses from different question types after injection in equal proportions, with the ratio order specified as logic, correctness, and hallucination. For example: ID 7 = 50% logic + 50% hallucination (a 2-type mix). ID 9: A distinct mixed variant where responses from all three error types (logic, correctness, hallucination) are combined to generate a single integrated type.

Effect of Different Error Types The results show that even without introducing specific error types, the model can still improve, but the improvement is mainly concentrated in three dimensions: MATH, BBH, and Truthfulness. When errors are introduced in the logic or correctness type, the performance in MATH, code-related tasks, and truthfulness will be enhanced. When introducing hallucination errors, the improvement in credibility is more significant compared to the previous two error types. We observe that the introduction of hallucinations is often associated with noticeably shorter model outputs, as illustrated in Appendix A.2. Reasoning tasks typically benefit from a step-by-step thought chain approach. Such short responses can adversely affect performance on complex reasoning tasks, where extended generation length is often necessary for success. Prior studies (Li et al., 2025; Guo et al., 2025) have emphasized that longer reasoning chains are positively correlated with improved performance on more challenging reasoning problems. These results demonstrate the necessity of introducing problem types and the influence of problem-type diversity on the comprehensive performance of multiple tasks. We present additional evaluation examples of error handling for the three types of baseline errors in Appendix A.2, demonstrating the effectiveness of SeaPO.

Effect of Mixing Strategy In Table 3, we conducted a preliminary exploration of different error type ratios of mixing. The underlying philosophical distinctions are reflected in the introduction of different error types and their varying impacts across tasks. All hallucination-related error types lead to substantial performance improvements on TruthfulQA, a benchmark specifically designed to evaluate sensitivity to hallucination.

Mixing error types improves overall performance: most average results rank top-2, and more

ID	Error Type	Math		Reasoning	Coding	Knowledge	Truthful		Avg.
		MATH	GSM	BBH	HumanEval	MMLU	MC1	MC2	
1	None (Instruct)	47.2	72.5	67.5	88.7	72.8	42.4	59.3	64.3
2	Untargeted	56.6	70.8	68.6	87.6	<u>72.9</u>	55.6	67.4	68.5
3	Logic	54.5	71.8	65.2	89.0	72.7	59.5	70.7	69.1
4	Correctness	<u>54.7</u>	69.6	65.4	<u>89.3</u>	72.8	57.8	68.8	68.3
5	Hallucination	53.7	74.4	66.7	<u>89.3</u>	72.5	60.8	<u>72.0</u>	69.9
6	3+4	52.9	**79.8**	<u>69.4</u>	87.7	**73.2**	52.6	64.9	68.6
7	3+5	52.5	77.5	67.4	<u>89.3</u>	72.6	**62.8**	**72.9**	**70.7**
8	4+5	53.9	75.7	67.7	88.0	72.7	<u>61.8</u>	71.7	<u>70.2</u>
9	Multi	55.8	<u>79.5</u>	**70.4**	**89.6**	**73.4**	50.2	64.7	69.1
10	3+4+5	**56.5**	75.4	67.0	88.6	72.5	60.2	70.9	<u>70.2</u>

Table 3: Results for different error types on each tasks. Training on the most frequent error type improves task performance. Math, code, and reasoning tasks, are prone to correctness and logical errors, while TruthfulQA, a knowledge-based task, is more susceptible to hallucinations. Constructing the relevant errors enhances performance in each task. The best results are bolded and the second best results are underlined.

Error Type	Qwen2.5-72B		Llama3.1-70B	
	Source	Injected	Source	Injected
Logic	11,249	**45,815**	9,891	**43,044**
Correctness	12,367	**43,577**	10,307	**38,894**
Hallucination	6,515	**47,069**	8,582	**46,995**

Table 4: Comparison of error types in the original answers and those after error injection. The number of specific error types in the modified answers increased compared to the original answers.

mixed types enhance nearly all subtasks (though less than single-task gains, e.g., GSM/BBH for ID 6/10). Results show that multi-error injection (ID 9) boosts scores (+4.8 avg, +9.0 MATH, +7.0 GSM), confirming effectiveness, though ID 10 outperforms. For generalization, we use ID 10 (full mixing). The introduction of specific errors led to notable improvements in the corresponding evaluation results. For instance, errors involving hallucination types, such as those in IDs 5, 7, 8, and 10, resulted in significant gains compared to cases without explicit hallucination injection (e.g., IDs 1 and 6), with the largest improvement nearing 10 points. When focusing on specific capabilities, such as mathematics and reasoning, a blend of logic and correctness is most effective. For truthfulness, combining logic with hallucination errors yields better results. In general, mixing a broader range of error types can lead to more comprehensive performance improvements across various tasks.

5.2 Analysis on Error-Injected Negative Sample Generation

Effect of Injected Error Quality To assess whether the model's error injection mechanism can generate responses with specific error types based on the original response, we conducted evaluations using models such as Llama3.1-70B and Qwen2.5-72B. The results are presented in Table 4. We employed these evaluation models to classify responses based on the question, reference answer, and rejection, performing binary classification to determine if the response exhibits a specific error type. The evaluation prompt is provided in Appendix A.1. The negative samples in the original dataset may not consistently exhibit greater error severity than their corresponding positive samples. In Table 4, our generated negative samples demonstrate a higher degree of incorrectness compared to the original set. The presence of less erroneous examples in the original negative samples likely hinders the effectiveness of preference training, potentially degrading overall model performance.

Table 5 presents a comparison of error correction performance across four severity levels (minor, moderate, major, critical) and three error categories (factual correctness, logical consistency, and hallucination). The evaluation is conducted on 1,000 sampled instances using two scoring methodologies: direct absolute scoring (Kim et al., 2024b) and relative win rate (Zheng et al., 2023), applied to Qwen2.5-72B-Instruct and DeepSeek V3. Notably, the performance gap becomes substantial only when the correction quality is critically poor,

	Qwen2.5-72B		DeepSeek-V3	
	Score	Win-Rate	Score	Win-Rate
UltraFeedback	6.87	-	7.09	-
Correctness	6.91	0.68	7.13	0.69
Logic	6.67	0.67	6.98	0.67
Hallucination	6.65	0.67	7.00	0.62
Minor	7.30	0.71	7.43	0.73
Moderate	7.05	0.70	7.22	0.67
Major	6.51	0.64	6.93	0.60
Critical	3.46	0.30	4.73	0.25

Table 5: Comparison of negative sample quality. The negative samples with error injection show no significant quality degradation compared to the original ones.

Performance	Baseline	Error Injector		
		1.5B	7B (Self)	72B
MATH	47.2	55.6	**56.5**	55.0
GSM	72.5	77.0	75.4	**83.5**
BBH	67.5	**67.6**	67.0	66.1
HumanEval	**88.7**	86.9	88.6	88.1
MMLU	72.8	**73.1**	72.7	72.4
Truthful-MC1	42.4	44.8	**60.2**	**60.2**
Truthful-MC2	59.3	59.8	70.9	**71.7**
Avg.	64.3	66.4	70.2	**71.0**

Table 6: Effect of Source of the Generated Error. Larger models have stronger error-injection capabilities.

Task	Baseline	Minor	Moderate	Major	Critical
MATH	47.2	**57.3**	56.7	56.6	53.3
GSM	**72.5**	64.4	67.6	70.8	62.1
BBH	67.5	67.7	68.3	68.6	**69.6**
HumanEval	**88.7**	88.2	87.7	87.6	86.0
MMLU	72.8	**73.2**	**73.2**	72.9	73.0
Truthful-MC1	42.4	49.8	52.3	**55.6**	45.7
Truthful-MC2	59.3	64.0	66.1	**67.4**	61.3
Avg.	64.3	66.4	67.4	**68.5**	64.4

Table 7: Comparison of the effects obtained by negative samples with different error levels. Models with the error level of "Major" work best, indicating that negative samples require recognizable errors.

with direct scores dropping from 6.87 to 3.46. Despite this, our proposed error-specific negative samples maintain performance comparable to original baselines across both evaluation methods and models, showing a direct score difference of no more than 1 point and a win rate difference of no more than 0.1. These results demonstrate that our negative samples are effective for model training without compromising output quality.

Effect of Source of the Generated Error We show in Table 6 the effect of different sizes (1.5B, 7B, and 72B) of error-injected negative samples of the same model architecture on the final training effect of the model, which shows that the models of different sizes have different abilities to introduce errors by prompt. This ability to introduce errors is relatively poor for small models and strong for large models. However, we find that even with small models, after error injection, the model's can all be further improved, and the gap between these scores is not very large, which indicates that SeaPO is not very demanding for the ability to correct errors, and further illustrates the robustness.

Instruct	RoN		SeaPO	
	DPO	KTO	DPO	KTO
64.3	64.8	65.7	69.4	**70.2**

Table 8: Impact of optimization objectives on average performance. "RoN" random selects one from three negative samples as the rejection (original UltraFeedback), while SeaPO selects from the rewritten negatives.

Effect of Severity of Errors Table 7 analyzes the impact of error severity on training, classifying introduced errors into four levels: Minor (few errors), Moderate (medium errors), Major (many errors), and Critical (almost entirely incorrect). Results show that prompting models to introduce errors generally improves performance via preference learning, except for Critical-level errors, where improvement is negligible. This highlights that negative samples must maintain a reasonable quality for the model to distinguish errors effectively. SeaPO achieves improvements by reducing the likelihood of negative samples: our negative samples contain more errors than original responses while maintaining comparable quality. Xu et al. (2024) can be categorized as introducing Major errors. Nevertheless, the performance resulting from these randomly injected errors remains inferior to that achieved through the introduction of targeted error types, as implemented in SeaPO.

5.3 Analysis on Error-Injected Preference Optimization

In Table 8, we present the average results across the five dimensions for both the data we constructed and the original data, evaluated under two optimization objectives. DPO is more susceptible to the influence of noisy data (Rafailov et al., 2023), where both positive and negative samples may contain correct responses, a problem that is particularly evident in the RoN (random selection of one of the remaining three samples using the original dataset). After training on the original data, the model performance showed slight volatility, while KTO demonstrated better generalization, with results showing a modest improvement. However, utilizing our generated, containing more injected errors, we observed an improvement in the DPO results (almost +4 points). When combined with KTO, which is less sensitive to sample quality, even better performance was achieved. This demonstrates the robustness of SeaPO in different learning algorithms.

6 Conclusion

Due to model limitations and no explicit quality control in preference data construction, positive and negative samples may have similar quality; in some cases, negative samples may even be better. To address this issue, we propose SeaPO that explicitly introduces strategically defined errors. Applying preference optimization reduces error probabilities and enhances overall performance. Evaluations across five capability dimensions and multiple models (1.5B-14B) demonstrate the robustness of our approach. These strategic errors vary depending on the task, and targeting the most error-prone types can significantly boost task performance.

Limitations

There are several limitations to our work. First, our experiments are constrained by available resources, particularly GPU memory capacity, which limits the number of models we can test, especially at larger scales (e.g., 70B models). Additionally, we did not define a broader range of error types, such as those related to grammar or safety, which could potentially enhance the performance of specific tasks. Another limitation is the lack of targeted error analysis for specific task types. As a result, the three error categories we defined may not fully capture less common error types that occur in certain tasks. Finally, our experiments were conducted using only one preference dataset. The quality of positive samples in different datasets may introduce variations in the final model results.

Ethics Statement

We take ethical considerations very seriously and strictly adhere to the ACL Ethics Policy. All experiments are conducted on open datasets and the findings and conclusions of this paper are reported accurately and objectively.

Acknowledgments

This work was supported in part by Guangdong S&T Program (Grant No. 2024B0101050003), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2024A1515011491), and Shenzhen Science and Technology Program (Grant Nos. ZDSYS20230626091203008, KJZD20231023094700001, KQTD20240729102154066). We would like to thank the anonymous reviewers and meta-reviewer for their insightful suggestions.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Zhiqiang Bao, Zihao Chen, Chang-Dong Wang, Wei-Shi Zheng, Zhenhua Huang, and Yunwen Chen. 2023. Post-distillation via neural resuscitation. *IEEE Transactions on Multimedia*, 26:3046–3060.

Lichang Chen, Shiyang Li, and Jun Yan. 2024a. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations*.

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

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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 arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2024. UltraFeedback: Boosting language models with high-quality feedback. In *Proceedings of the 41st International Conference on Machine Learning*.

Hexuan Deng, Wenxiang Jiao, Xuebo Liu, Min Zhang, and Zhaopeng Tu. 2024. Newterm: Benchmarking real-time new terms for large language models with annual updates. In *Advances in Neural Information Processing Systems*, volume 37, pages 35760–35795. Curran Associates, Inc.

Qingxiu Dong, Li Dong, Xingxing Zhang, Zhifang Sui, and Furu Wei. 2025. Self-boosting large language models with synthetic preference data. In *The Thirteenth International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. 2024. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*.

Guangzeng Han, Weisi Liu, and Xiaolei Huang. 2025. Attributes as textual genes: Leveraging llms as genetic algorithm simulators for conditional synthetic data generation. *Preprint*, arXiv:2509.02040.

Guangzeng Han, Jack Tsao, and Xiaolei Huang. 2024. Length-aware multi-kernel transformer for long document classification. In *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, pages 278–290, Mexico City, Mexico. Association for Computational Linguistics.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, Miami, Florida, USA. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shijia Huang, Jianqiao Zhao, Yanyang Li, and Liwei Wang. 2023. Learning preference model for LLMs via automatic preference data generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9187–9199, Singapore. Association for Computational Linguistics.

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Dongyoung Kim, Kimin Lee, Jinwoo Shin, and Jaehyung Kim. 2024a. Aligning large language models with self-generated preference data. *arXiv preprint arXiv:2406.04412*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024b. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loka Li, Zhenhao Chen, Guangyi Chen, Yixuan Zhang, Yusheng Su, Eric Xing, and Kun Zhang. 2024. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *arXiv preprint arXiv:2402.12563*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. From system 1 to system 2: A survey of reasoning large language models. *Preprint*, arXiv:2502.17419.

Yiming Liang, Ge Zhang, Xingwei Qu, Tianyu Zheng, Jiawei Guo, Xinrun Du, Zhenzhu Yang, Jiaheng Liu, Chenghua Lin, Lei Ma, et al. 2024. I-sheep: Self-alignment of llm from scratch through an iterative self-enhancement paradigm. *arXiv preprint arXiv:2408.08072*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *arXiv preprint arXiv:2304.03277*.

Shuhan Qi, Zhengying Cao, Jun Rao, Lei Wang, Jing Xiao, and Xuan Wang. 2023. What is the limitation of multimodal llms? a deeper look into multimodal llms through prompt probing. *Information Processing & Management*, 60(6):103510.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. 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*.

Jun Rao, Liang Ding, Shuhan Qi, Meng Fang, Yang Liu, Li Shen, and Dacheng Tao. 2023a. Dynamic contrastive distillation for image-text retrieval. *IEEE Transactions on Multimedia*, pages 1–13.

Jun Rao, Zepeng Lin, Xuebo Liu, Xiaopeng Ke, Lian Lian, Dong Jin, Shengjun Cheng, Jun Yu, and Min Zhang. 2025. APT: Improving specialist LLM performance with weakness case acquisition and iterative preference training. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20958–20980, Vienna, Austria. Association for Computational Linguistics.

Jun Rao, Xuebo Liu, Lian Lian, Shengjun Cheng, Yunjie Liao, and Min Zhang. 2024. CommonIT: Commonality-aware instruction tuning for large language models via data partitions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10064–10083, Miami, Florida, USA. Association for Computational Linguistics.

Jun Rao, Xv Meng, Liang Ding, Shuhan Qi, Xuebo Liu, Min Zhang, and Dacheng Tao. 2023b. Parameter-efficient and student-friendly knowledge distillation. *IEEE Trans. Multim.*, pages 1–12.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi 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. 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*.

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*.

Kaiwen Wei, Jiang Zhong, Hongzhi Zhang, Fuzheng Zhang, Di Zhang, Li Jin, Yue Yu, and Jingyuan Zhang. 2025. Chain-of-specificity: Enhancing task-specific constraint adherence in large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2401–2416, Abu Dhabi, UAE. Association for Computational Linguistics.

Weiwei Xiao, Yongyong Chen, Qiben Shan, Yaowei Wang, and Jingyong Su. 2024. Feature distribution matching by optimal transport for effective and robust coreset selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. 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*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6268–6278, Singapore. Association for Computational Linguistics.

Kaishuai Xu, Tiezheng Yu, Wenjun Hou, Yi Cheng, Chak Tou Leong, Liangyou Li, Xin Jiang, Lifeng Shang, Qun Liu, and Wenjie Li. 2024. Subtle errors matter: Preference learning via error-injected self-editing. *arXiv preprint arXiv:2410.06638*.

Shunzhi Yang, Jinfeng Yang, MengChu Zhou, Zhenhua Huang, Wei-Shi Zheng, Xiong Yang, and Jin Ren. 2024a. Learning from human educational wisdom: A student-centered knowledge distillation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4188–4205.

Shuo Yang, Zheyu Zhang, Bardh Prenkaj, and Gjergji Kasneci. 2025. Doubling your data in minutes: Ultra-fast tabular data generation via llm-induced dependency graphs. *Preprint*, arXiv:2507.19334.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024b. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Zheyu Zhang, Shuo Yang, Bardh Prenkaj, and Gjergji Kasneci. 2025. Not all features deserve attention: Graph-guided dependency learning for tabular data generation with language models. *Preprint*, arXiv:2507.18504.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024a. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Siyan Zhao, John Dang, and Aditya Grover. 2024b. Group preference optimization: Few-shot alignment of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xinping Zhao, Dongfang Li, Yan Zhong, Boren Hu, Yibin Chen, Baotian Hu, and Min Zhang. 2024c. SEER: Self-aligned evidence extraction for retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3027–3041, Miami, Florida, USA. Association for Computational Linguistics.

Xinping Zhao, Jindi Yu, Zhenyu Liu, Jifang Wang, Dongfang Li, Yibin Chen, Baotian Hu, and Min Zhang. 2024d. Medico: Towards hallucination detection and correction with multi-source evidence fusion. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 34–45, Miami, Florida, USA. Association for Computational Linguistics.

Xinping Zhao, Yan Zhong, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Dongfang Li, Baotian Hu, and Min Zhang. 2025. FunnelRAG: A coarse-to-fine progressive retrieval paradigm for RAG. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3029–3046, Albuquerque, New Mexico. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. 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*.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. 2024. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*.

A Appendix

A.1 Prompts

Multiple experiments, specifically, the results in Table 3 and the Moderate and Major error categories in Table 6, demonstrate that, despite the simplicity and diversity of the prompt construction (including both prompts with specific error types and those without), our method consistently yields significant performance gains. In several tasks across different model architectures, improvements of up to 5 percentage points are observed.

We provide the prompt templates used in our study. The first template, titled *Prompt Template for Error-Injected Negative Sample Generation*, is used for error injection. This template directs models to deliberately modify a given response in order to introduce a specified error type. It includes fields for the question, original response, error type, and error description. Models are then instructed to revise the response intentionally to incorporate the error, without providing additional explanation. The second template, titled *Prompt Template for Error Detection*, is used for error detection. This template directs evaluators to determine if a given answer contains the specified error. Evaluators are provided with the question, answer, error type, and error description, and are required to respond with "yes" or "no" based on the presence of the error, without further elaboration. The third template, *Prompt Template for Error Injection*, is used to generate responses containing injected errors. The fourth template, *Prompt Template for Error Injection by Severity Level*, is designed to produce prompts corresponding to three levels of error severity: Minor (Level 1), Moderate (Level 2), and Major (Level 3), without specifying error types. For prompts related to evaluating the quality of generated responses, refer to the "Prompt Scoring" and "Prompt Comparison sections".

A.2 Case Studies

Error Injection Cases The following examples illustrate error injection in three distinct cases.

For the correctness case, the statement "Exams are the only way to measure a person's knowledge" is factually incorrect because it disregards valid alternative assessment methods, such as course grades, project outcomes, and portfolio reviews. The claim that exams are the sole tool for evaluation contradicts established practices in educational assessment, making the statement inaccurate. This error arises from providing information that does not align with factual knowledge.

For the logic case, the misinterpretation of Thoreau's metaphor becomes apparent when the response states that someone cannot keep pace because they "cannot hear the same drummer." The original metaphor emphasizes individuality ("hearing a different drummer"), but this interpretation shifts the meaning to one of auditory limitation. By altering the intended message, the response loses coherence and fails to preserve the logical structure of the argument. Such errors occur when the reasoning process distorts or misrepresents the relationships between ideas.

For the hallucination case, the fabrication of fictional concepts like "tiny invisible fairies" and "fairy theory" introduces imaginary elements with no basis in reality or historical context. Associating these fictitious ideas with Descartes' era, despite the absence of any scientific or historical grounding, demonstrates how the model generates content that does not exist. This type of error results from inventing concepts or details entirely unsupported by evidence, leading to responses that include hallucinatory content.

Evaluation Cases Based on the evaluation results, we present case studies to compare the baseline model (Qwen2.5-7B-Instruct) with our fine-tuned model across three error categories: correctness, logic, and hallucination. In each case, the baseline answer refers to the original model's output, while our answer represents the improved model's response. Incorrect components of the baseline responses are highlighted in red, while the corrected sections of our model's responses are also marked in red. These cases illustrate the reduction in errors achieved by our fine-tuned model compared to the baseline.

For correctness errors, the baseline model incorrectly applied the order of operations, leading to an erroneous result. Our model rectified this by adhering to the correct sequence of calculations.

For logic errors, the baseline model erroneously calculated the interest for a single phone and propagated this mistake to the final result. Our model identified and addressed the reasoning flaw.

For hallucination errors, the baseline model misinterpreted the definition of the hypotenuse's height, introducing incorrect information. Our model corrected this by applying domain-specific knowledge.

Prompt Template for Error-Injected Negative Sample Generation

Your task is to deliberately modify the provided response to introduce the specified error.

Task:
Analyze the given question, original response, error type, and error description. Then, revise the response to intentionally include the specified error.

1. **Question**: {question}
2. **Original Response**: {response}
3. **Error Type**: {error_type}
4. **Error Description**: {error_description}

Instructions:
- Modify the original response to clearly incorporate the specified error.
- Do not include any explanations, notes, or other text in your output.
- Output only the revised response.

Revised response:

Prompt Template for Error Detection

You are an AI evaluator. I will give you a question, its answer, and a specific error type with a description. Your task is to determine if the answer contains the described error. Please respond with "yes" if the answer contains the error and "no" if it does not. Only respond with "yes" or "no" without any further explanation.

Question: "{question}"
Answer: "{answer}"
Error Type: "{error_type}"
Error Description: "{error_description}"

Does the answer contain the described error?

Prompt Template for Generation of Error-Injection

Please help me design a prompt in Python string format. I will input the question, its correct answer, an error type, and a brief description of that error type. The model needs to intentionally modify the correct answer to incorporate the specified error type, outputting only the modified incorrect answer without any extra content. The Python string prompt should include placeholders for the input information.

Generation of Error-Injection Prompts of Different Levels

Please help me design a prompt in the form of a Python string. I will input a question, its correct answer, an error severity level, and a brief description of that severity level. The model needs to intentionally modify the correct answer into an incorrect one that matches the specified error severity. The output should only contain the modified incorrect answer, with no additional content. The Python string prompt must include placeholders for the input information.

Prompt Scoring

Task: Evaluate the quality of an answer while COMPLETELY IGNORING factual errors.
Note: The answer may contain inaccuracies or technical errors. DO NOT consider correctness in scoring.
Dimensions (1-10 scale): 1. Logical Structure - Organization and reasoning flow 2. Language Fluency - Grammar and readability 3. Topic Relevance - Alignment with question intent 4. Depth - Detail level and analysis complexity 5. Consistency - Self-contradiction check
Input: Question: question Answer: answer
Output Format: Single integer between 1-10 calculated as: (Round average of all dimension scores to nearest integer)
ONLY OUTPUT THE FINAL INTEGER

Prompt Comparison

Compare two answers (A/B) while COMPLETELY IGNORING FACTUAL ERRORS. Determine which response demonstrates superior quality in non-factual dimensions.
Critical Rules: 1. DO NOT CONSIDER: Accuracy, technical correctness, or truthfulness 2. EVALUATE ONLY: Structural integrity of arguments Language mechanics and readability Relevance to question intent Depth of analysis Internal consistency 3. Tie only if indistinguishable across all dimensions
Assessment Protocol: 1. Read both answers fully 2. Identify potential errors → Explicitly exclude from evaluation 3. Compare dimension-by-dimension: a) Logical Structure: Which has clearer reasoning flow? b) Language Fluency: Which is more professionally written? c) Topic Relevance: Which stays focused on core question? d) Depth: Which provides more layered analysis? e) Consistency: Which avoids self-contradictions? 4. Declare winner with majority advantages
Input: Question: {question} Answer A: {answer_a} Answer B: {answer_b}
Output: Strictly ONE of these options: A | B | tie
DO NOT OUTPUT ANY OTHER TEXT OR EXPLANATION.

Case of Correctness Error Injection

Question:
Check if the following statement is true or false:
Exams are the best way to measure a person’s knowledge

Correct answer:
False. Exams are not the only way to measure a person’s knowledge. Other ways include course grades, project outcomes, problem-solving tasks, interview questions, portfolio reviews, and letters of recommendation. Exams are one tool used to measure knowledge, but they should not be the only tool relied on.

Answer injected with correctness error:
False. Exams are the only way to measure a person’s knowledge. Other methods such as course grades, project outcomes, problem-solving tasks, interview questions, portfolio reviews, and letters of recommendation are invalid. Exams are the definitive tool for assessing knowledge.

Case of Logic Error Injection

Question:
Thoreau's metaphors are highly visual. Though they're clever and original, they aren't far-fetched. Thoreau takes his comparisons from nature and from other things with which he and his audience are familiar. To be sure you understand Thoreau's figures of speech, paraphrase the following metaphor. Do his metaphors apply to readers today?
"If a man does not keep pace with his companions, perhaps it is because he hears a different drummer. Let him step to the music which he hears, however measured or far away."

Correct answer:
In this metaphor, Thoreau is suggesting that if a person doesn't conform or fit in with others around them, it might be because they have their own unique path and aspirations. He encourages the individual to follow their own internal guidance and passions, no matter how different or distant it may seem from the norm.
Thoreau's metaphors can still apply to readers today, as it highlights the importance of individuality and staying true to oneself in a world that often values conformity.

Answer injected with logic error:
In this metaphor, Thoreau is suggesting that if a person doesn't keep pace with others, it is because they are deaf to the same drummer. He encourages the individual to follow their own internal guidance and passions, no matter how different or distant it may seem from the norm.

Case of Hallucination Error Injection

Question:
I am refering to earlier, back in Descartes age
"If a man does not keep pace with his companions, perhaps it is because he hears a different drummer. Let him step to the music which he hears, however measured or far away."

Correct answer:
Back in Descartes' age, there was an idea that the physical world was composed of atoms that were arranged in a certain way and that this arrangement formed the foundation of all knowledge. This is now known as the atomic theory and it is one of the core principles of modern science.

Answer injected with hallucination error:
Back in Descartes' age, there was an idea that the physical world was composed of tiny invisible fairies that danced in a certain way and that this dance formed the foundation of all knowledge. This is now known as the fairy theory and it is one of the core principles of modern science.

Case of Correctness Errors in Evaluation

Question:
Find $\frac{1}{-6} \cdot 6 \cdot 7 + 8 \cdot 2 \div 8 \cdot (7 - 1)$.

Baseline answer:
First, we simplify the expression inside the parentheses: $7 - 1 = 6$. Next, we perform the multiplication and division from left to right:

$$\frac{1}{-6} \cdot 6 \cdot 7 = -1 \cdot 7 = -7,$$

and

$$8 \cdot 2 \div 8 = 16 \div 8 = 2.$$

Then, we multiply the results of the two parts:

$$-7 \cdot 2 = -14.$$

Finally, we add the result to the remaining part of the expression:

$$-14 + 8 \cdot 2 = -14 + 16 = 2.$$

Therefore, the final answer is $\boxed{2}$. Final Answer: The final answer is 2. I hope it is correct.

Our answer:
First, we simplify the expression inside the parentheses: $7 - 1 = 6$. Then, we simplify the division and multiplication: $8 \cdot 2 \div 8 = 2$. So, the expression becomes

$$\frac{1}{-6} \cdot 6 \cdot 7 + 2 \cdot 6 = -7 + 12 = \boxed{5}.$$

Final Answer: The final answer is 5. I hope it is correct.

Case of Logic Errors in Evaluation

Question:
Shiela bought five cell phones for \$150 each for a 3-month installment. A 2% interest will be charged for each unit. How much will Shiela pay each month for 3 months?

Baseline answer:
First, we need to calculate the total cost of the cell phones before interest. 5 cell phones at \$150 each is 5 x 150 = \$750. The interest for each cell phone is 2%, so for 5 cell phones, the total interest is 5 x 750 x 0.02 = \$75. The total cost including interest is 750 + 75 = \$825. Since she is paying this over 3 months, we divide the total cost by 3: 825 / 3 = \$275. So the answer is 275. Each month, Shiela will pay \$275.

Our answer:
The total cost of the cell phones is 5 x \$150 = \$750. The interest for each unit is 2% of \$150, which is 0.02 x \$150 = \$3. So the total interest for all the cell phones is 5 x \$3 = \$15. The total amount to be paid is \$750 + \$15 = \$765. The monthly payment for 3 months is \$765 / 3 = \$255. So the answer is \$255.

Case of Hallucination Errors in Evaluation

Question:
In a certain isosceles right triangle, the altitude to the hypotenuse has length $4\sqrt{2}$. What is the area of the triangle?

Baseline answer:
In an isosceles right triangle, the altitude to the hypotenuse splits the triangle into two smaller isosceles right triangles and also bisects the hypotenuse. Let the legs of the original triangle be of length x. The hypotenuse, then, is $x\sqrt{2}$. The altitude to the hypotenuse in an isosceles right triangle is also the geometric mean of the segments it divides the hypotenuse into, which are each $\frac{x\sqrt{2}}{2}$. Therefore, the altitude h can be expressed as:

$$h = \frac{x\sqrt{2}}{2} \cdot \frac{x\sqrt{2}}{2} = \frac{x^2}{2}$$

Given that the altitude is $4\sqrt{2}$, we have:

$$\frac{x^2}{2} = 4\sqrt{2}$$

Multiplying both sides by 2, we get:

$$x^2 = 8\sqrt{2}$$

Taking the square root of both sides, we find:

$$x = \sqrt{8\sqrt{2}} = 2\sqrt{2\sqrt{2}}$$

The area A of the isosceles right triangle is given by:

$$A = \frac{1}{2}x^2 = \frac{1}{2} \cdot 8\sqrt{2} = 4\sqrt{2}$$

Therefore, the area of the triangle is $\boxed{16}$. Final Answer: The final answer is 16. I hope it is correct.

Our answer:
Let the legs of the isosceles right triangle be x. Then, the hypotenuse is $x\sqrt{2}$. The altitude to the hypotenuse is $\frac{x\sqrt{2}}{2}$, since the altitude of an isosceles right triangle is half the length of the hypotenuse. We are given that the altitude is $4\sqrt{2}$, so we have

$$\frac{x\sqrt{2}}{2} = 4\sqrt{2} \Rightarrow x\sqrt{2} = 8\sqrt{2} \Rightarrow x = 8.$$

The area of the triangle is $\frac{1}{2}x^2 = \frac{1}{2}(8^2) = \boxed{32}$. Final Answer: The final answer is 32. I hope it is correct.