Stream Aligner: Efficient Sentence-Level Alignment via Distribution Induction

Hantao Lou[1,2], **Jiaming Ji**[1,2], **Kaile Wang**[1,2], **Yaodong Yang**[1,2,†]

[1] Institute for AI, Peking University
[2] State Key Laboratory of General Artificial Intelligence, Institute for AI, Peking University
{hantaolou.htlou, jiamg.ji}@gmail.com
wkl@stu.pku.edu.cn
yaodong.yang@pku.edu.cn

Abstract

The rapid advancement of large language models (LLMs) has led to significant improvements in their capabilities, but also to increased concerns about their alignment with human values and intentions. Current alignment strategies, including adaptive training and inference-time methods, have demonstrated potential in this area. However, these approaches still struggle to balance deployment complexity and capability across various tasks and difficulties. In this work, we introduce the Streaming Distribution Induce Aligner (*Stream Aligner*), a novel alignment paradigm that combines efficiency with enhanced performance in various tasks throughout the generation process. *Stream Aligner* achieves dynamic sentence-level correction by using a small model to learn the preferences of the suffix sentence, iteratively correcting the suffix sentence output by the upstream model, and then using the corrected sentence to replace the suffix sentence in subsequent generations. Compared to *Aligner*, our experiments demonstrate that *Stream Aligner* reduces reliance on the capabilities of additional models, enhances the reasoning abilities of LLMs, and decreases latency during user interaction. Specifically, *Stream Aligner*-2B model has achieved a maximum improvement of 41.2% in helpfulness, 36.0% in harmlessness on the tested Llama2-70B-chat model, and *Stream Aligner*-8B has achieved an improvement of 3.5% on the math ability of the tested Llama3-70B-Instruct model.

1 Introduction

Large language models (LLMs) can perform various downstream tasks (Touvron et al. 2023; Achiam et al. 2023), but they may exhibit unintended behaviors (Ji et al. 2024c; Hubinger et al. 2024). The alignment of LLMs aims to ensure the behaviours of LLMs are consistent with human intention and value (Ji et al. 2023, 2024d). As LLMs continue to scale up in size and capability, the need for lightweight, model-agnostic, yet efficient alignment methods becomes increasingly critical.

Currently, training methods such as supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022; Taori et al. 2023; Ji et al. 2024d) are the most widely recognized approaches to alignment (Bai et al. 2022a; Rafailov et al. 2024; Bai et al. 2022b; Dai et al. 2023). However, as the scale of LLMs increases, these training methods also face issues of rising data requirements, computational power consumption (Ding et al. 2023), and extremely sensitive to parameters and training data, especially in reasoning-related tasks (Casper et al. 2023).

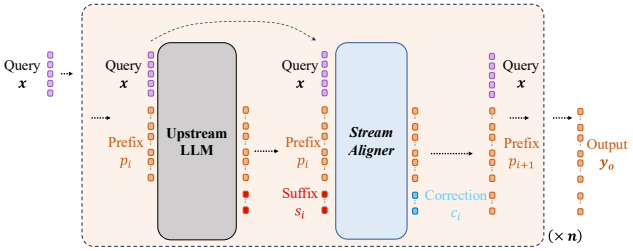


Figure 1: **Operational Dynamics of the *Stream Aligner* Generation Pipeline.** *Stream Aligner* serves as an plug-and-play module in the generation pipeline. It corrects the sentence generated by the upstream model and then feeds the corrected suffix back to the upstream model for further generation until the end of responses. This pipeline ensures every sentence in the output is aligned with the *Stream Aligner* model, thereby aligned with the human preference.

Inference-time methods, including speculative sampling (Chen et al. 2023), and safe-decoding (Xu et al. 2024), refine the inference algorithms to alter the text generation mechanisms of LLMs. These adjustments seek to align the responses and better elicit the latent knowledge of LLMs in a manner that avoids the expense and time commitment of additional training (Paul, Ajeya, and Xu 2024). Inference-time methods share the advantage of lightweight and deploying convenience, but they generally struggle to precisely distill human value and intent into the LLMs outputs in long context generation since a single token output cannot carry a complete unit of semantics (Ji et al. 2024a).

Meanwhile, the approach of incorporating additional models to base models has been drawing much attention recently. These approaches aim to distill human preferences within a smaller additional model, which is later incorporated into the predefined generation pipeline along with the targeted upstream model, as demonstrated by works like *Aligner* (Ji et al. 2024a; Yang et al. 2024). These methods can achieve

[0]† Corresponding author.

outstanding performances in areas such as value alignment and multi-objective alignment. However, these methods also have certain drawbacks: they are not able to fully elicit the latent knowledge of the upstream model, leading to a high dependency on the capabilities of the additional models in tasks related to model capability, thus making it difficult to achieve good performance; at the same time, these methods have relatively high latency, which affects user experience. This brings about a further need: *How can we elicit the capabilities of upstream models according to human preferences in the inference process?*

In this work, we combine the advantage of inference-time strategies and additional models to propose *Streaming Distribution Induce Aligner* (*Stream Aligner*), a sentence-level correction mechanism that stimulates the potential of the base model while conserving the distillation of human preferences. This is achieved by narrowing down the correction scope of *Aligner* to sentence level, feeding the corrected output back to the base model, and repeating this process. Specifically, *Stream Aligner* is fine-tuned on a preference dataset to learn the residuals of preferred and non-preferred last sentences under a fixed prompt and answer prefix. It is then integrated into the generation cycle depicted in Figure 1, continuously correcting sentences generated by the upstream model and incorporating them into the prefix to achieve sentence-level alignment. Compared to *Aligner*'s single-round generation, *Stream Aligner*'s paradigm has the following advantages:

- **Reduced dependency on additional model capabilities** *Stream Aligner* achieves distribution induction through sentence-level correction, thereby leveraging more of the performance of the upstream model and reducing dependence on the size and scale of the additional model. Specifically, we apply the *Stream Aligner* 2B model to correct the Llama2-70B-chat model, resulting in an increase in the helpfulness of responses by 41.2% and an increase in the harmlessness of responses by 36.0%, which is significantly higher than the results of *Aligner*-7B.
- **Enhanced reasoning abilities through step-by-step correction** In tasks related to reasoning, *Stream Aligner*'s distribution induction is observed to correct the incorrect part of the reasoning process in the upstream model and add inductions to the correct answer for subsequent steps, thereby enhancing the model's reasoning abilities. For a detailed comparison between Stream Aligner and the classic generation method, please refer to Figure 2. The experiments show that the longer the average intervention by *Stream Aligner* on the test set, the higher the accuracy after the intervention, as shown in Figure 3.

2 Formulation of Stream Paradigm

Preliminary: SFT and the *Aligner* Paradigm

Supervised Fine-tuning(SFT) SFT aims to fine-tune pre-trained LLM through supervised learning to generate target answers. For a high-quality dataset $\mathcal{D}_{\text{SFT}} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$, the SFT objective is to obtain a model $\pi_{\boldsymbol{\theta}}^{\text{SFT}}$ to minimize the negative log-likelihood loss:

$$\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{D}_{\text{SFT}}\right) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{\text{SFT}}}\left[\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\right]. \quad (1)$$

Algorithm 1: *Stream Aligner* Module

Require: Sentence-level preference dataset $\mathcal{D}$ where it contains $\{\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{y}_i^1, \boldsymbol{y}_i^2\}_{i=1}^n$; pre-train model $\mathcal{M}$; upstream model $\mathcal{M}_{\mathcal{B}}$; prompt dataset $\mathcal{D}_{\text{q}}$:$\{\boldsymbol{q}_i\}_{i=1}^n$
Ensure: Generated dataset $\mathcal{D}_{\text{full}}$:$\{\boldsymbol{q}_i, \boldsymbol{a}_i\}_{i=1}^n$
Initialize model $\mathcal{A}$ with weights from $\mathcal{M}$
Stage1: *Stream Aligner* Training
for each epoch **do**
  for each $(\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{y}_i^1, \boldsymbol{y}_i^2) \in \mathcal{D}$ **do**
    $\hat{\boldsymbol{y}}_i \leftarrow \mathcal{A}$.generate$(\boldsymbol{q}_i + \boldsymbol{p}_i + \boldsymbol{y}_i^1)$
    $\boldsymbol{\theta}_{\mathcal{A}} \leftarrow \boldsymbol{\theta}_{\mathcal{A}} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i^2)$
  end for
end for
Stage2: *Stream Aligner* Inference
for each $\boldsymbol{q}_i \in \mathcal{D}_{\text{q}}$ **do**
  Initialize $\boldsymbol{p}_i \leftarrow \varnothing$
  while True **do**
    $\boldsymbol{y}_i^1 \leftarrow \mathcal{M}_{\mathcal{B}}$.generate$(\boldsymbol{q}_i + \boldsymbol{p}_i)$
    $\boldsymbol{y}_i^2 \leftarrow \mathcal{A}$.generate$(\boldsymbol{q}_i + \boldsymbol{p}_i + \boldsymbol{y}_i^1)$
    $\boldsymbol{p}_i \leftarrow$ concatenate$(\boldsymbol{p}_i, \boldsymbol{y}_i^2)$
    if $\boldsymbol{y}_i^2 = \varnothing$ or $|\boldsymbol{p}_i \geq$ max_length$|$ **then**
      break;
    end if
  end while
  Acquire final answer $\boldsymbol{a}_i = \boldsymbol{p}_i$
  $\mathcal{D}_{\text{full}}$.append$(\boldsymbol{q}_i, \boldsymbol{a}_i)$
end for

The *Aligner* Paradigm The *Aligner* (Ji et al. 2024a) fine-tunes the model based on a preference dataset $\mathcal{M}$ to learn the correction residuals between preferred and non-preferred responses. For a dataset $\mathcal{M} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}_o^{(i)}, \boldsymbol{y}_c^{(i)}\}_{i=1}^N$, where $\boldsymbol{x}$ represents the user's query, $\boldsymbol{y}_o$ is the original answer, and $\boldsymbol{y}_c$ is the corrected answer according to established principles, *Aligner* is a conditional seq2seq model parameterized by $\boldsymbol{\phi}$, denoted as $\mu_{\boldsymbol{\phi}}(\boldsymbol{y}_c|\boldsymbol{y}_o, \boldsymbol{x})$. The model reassigns the preliminary answer $\boldsymbol{y}_o$ to the aligned answer $\boldsymbol{y}_c$. The training objective of *Aligner* is to minimize the following loss:

$$\mathcal{L}_{\textit{Aligner}}(\boldsymbol{\phi}, \mathcal{M}) = -\mathbb{E}_{\mathcal{M}}\left[\log \mu_{\boldsymbol{\phi}}(\boldsymbol{y}_c|\boldsymbol{y}_o, \boldsymbol{x})\right]. \quad (2)$$

Training and Inference Pipeline of *Stream Aligner*

Compared to *Aligner*, *Stream Aligner* refines the original correction process by correcting each sentence step by step, thereby improving the accuracy of the correction paradigm. The following describes the specific training and inference process of *Stream Aligner*. The entire pipeline of the *Stream Aligner* algorithm is shown in Algorithm 1.

***Stream Aligner* Model Training** *Stream Aligner* learns the residuals between preferred and non-preferred responses through a sentence-level preference dataset $\mathcal{D}$. For a sentence-level preference dataset $\mathcal{D} = \{\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{y}_i^1, \boldsymbol{y}_i^2\}_{i=1}^n$, where $\boldsymbol{q}$ represents the user's query, $\boldsymbol{p}$ is the common response prefix of $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$, and $\boldsymbol{y}^1$ is the original answer while $\boldsymbol{y}^2$ is the corrected answer according to established principles, the *Stream Aligner* model, parameterized by $\boldsymbol{\theta}$ and denoted as $\mathcal{A}$, reduces the residual between $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ conditioned on the

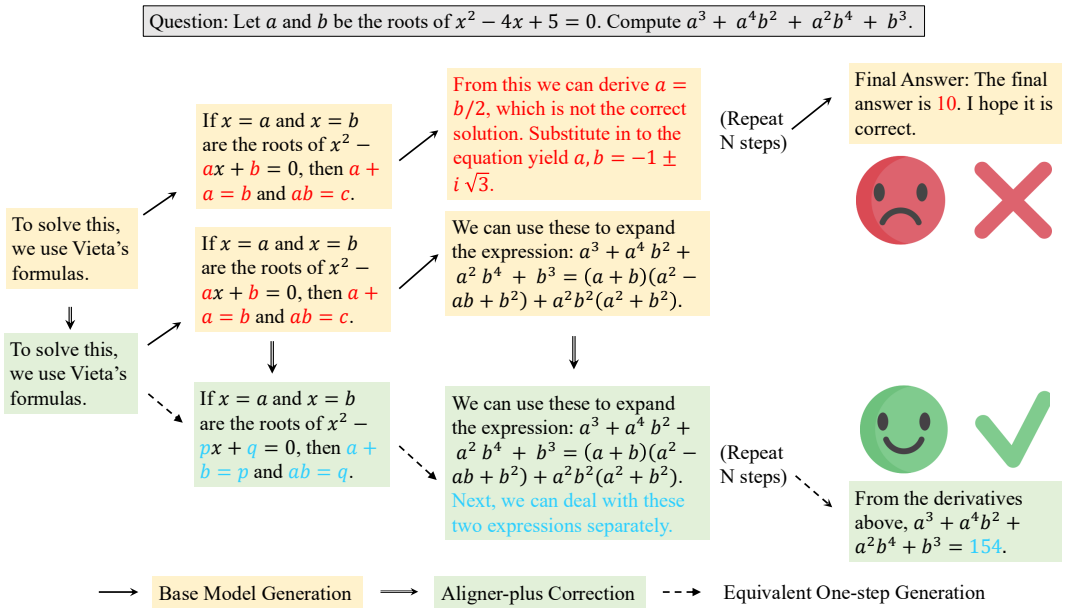


Figure 2: **Comparative Demonstration of the *Stream Aligner* Module.** Typically, the *Stream Aligner* has two working patterns depending on the original suffix sentence: (i) If the suffix is correct, copy it; (ii) If the suffix is incorrect or unsatisfying, rewrite it to make it correct and better. In this way, *Stream Aligner* can eliminate minor mistakes and toxic output made by the upstream model, thereby eliciting more correct latent knowledge and reducing the reliance on *Stream Aligner* capability.

question $\boldsymbol{q}$ and the prefix $\boldsymbol{p}$. The training objective of *Stream Aligner* is to minimize the following loss:

$$\mathcal{L}_{\textit{Stream Aligner}}(\boldsymbol{\theta}, \mathcal{D}) = -\mathbb{E}_{\mathcal{D}}\left[\log \mathcal{A}(\boldsymbol{y}^2|\boldsymbol{y}^1, \boldsymbol{q}+\boldsymbol{p})\right]. \quad (3)$$

***Stream Aligner* Model Inference** During the inference process of *Stream Aligner*, the *Stream Aligner* takes the question and the prefix $\boldsymbol{q}+\boldsymbol{p}$ as input, where $\boldsymbol{p}$ is initialized as $\varnothing$, and the upstream model $\mathcal{M}_{\mathcal{B}}$ generates the original answer $\boldsymbol{y}^1$ step by step. The trained *Stream Aligner* model then corrects this answer with $\boldsymbol{y}^2$. Each generated correction is incorporated into the prefix $\boldsymbol{p}$ until the generation stops or the prefix exceeds the maximum length. The final answer to the question $\boldsymbol{q}$ is the resulting prefix $\boldsymbol{p}$.

3 Experiments

In this section, we assess the effectiveness of *Stream Aligner* in three evaluation metrics: helpful and harmless QA, math questions, and summary tasks. We further analyze the evaluation results and do an ablation study on these situations.

Experiment Setup

Dataset We use different datasets for each task: HH-RLHF (Bai et al. 2022a) for helpful and harmless QA, and MATH (Hendrycks et al. 2021) for math and reasoning tasks [1]. Considering that no current dataset constructs a fine-grained preference in these two datasets, we create two additional datasets based on the prompts given in these two datasets. We use Alpaca-7B (Taori et al. 2023), Llama2-(7B, 70B)-Chat (Touvron et al. 2023), Llama3-(8B, 70B)-Instruct (Meta 2024) as upstream models to generate original answer sentences, and we take GPT-4 (Achiam et al. 2023), Llama3-70B-Instruct, Qwen1.5-110B-Chat (Team 2024) as annotators to refine the suffix sentences after generation. These refinements were conducted under a well-written prompt demonstrating the constraint and principles of our correction paradigm, which we expect the *Stream Aligner* to learn from: Rewrite the bad, improve the neutral, and keep the good.

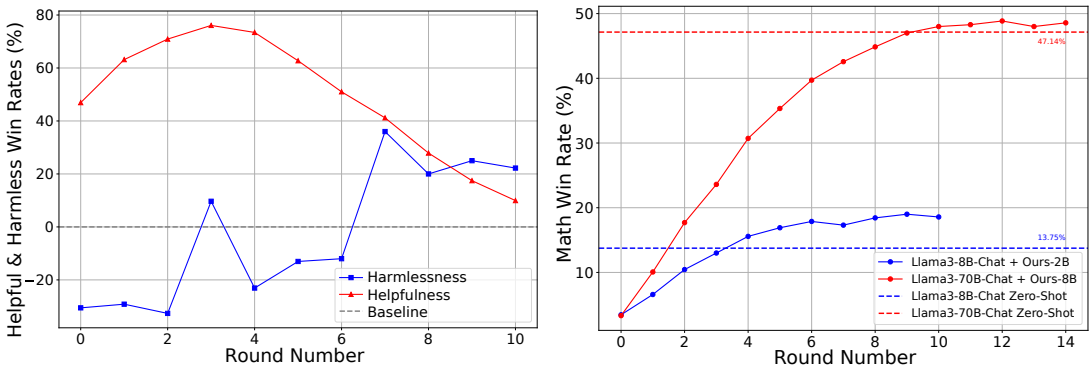
Models We train *Stream Aligner*-(2B, 8B) models based on Gemma1.1-2B (Team et al. 2024) and Llama3-8B foundation models using the dataset above. We then incorporate them into the deploying pipeline described in Section 2, with Llama3-(8B,70B)-Instruct as upstream models.

Evaluation Metrics Our evaluation metrics vary on different tasks, but the core idea remains the same: Compete the *win rate* with the direct answer of the upstream model. We sample a test set from BeaverTails (Ji et al. 2024b) and MATH for evaluation. We directly generate answers from upstream models and use the *Stream Aligner* pipeline to generate a corrected answer. We then utilize powerful models (in our case, GPT-4 for helpful & harmless QA and math) to evaluate the two answers and took the win rate of *Stream Aligner* against the upstream model as the evaluation result.

Experiment Results

The performance of *Stream Aligner* pipeline on math and helpful & harmless QA tasks is shown in Figure 3. Under these tasks, we can observe an outstanding improvement in

[1] We choose these two distinct tasks to prove the effectiveness of *Stream Aligner* on both QA tasks and reasoning tasks.



(a) (b)

Figure 3: Performance of *Stream Aligner* models. (a) The win rate of Llama2-70B-Chat + *Stream Aligner*-2B on helpfulness and harmlessness, compared to the baseline generated by Llama2-70B-Chat. (b) The win rate (accuracy) of Llama3-8B-Instruct + *Stream Aligner*-2B and Llama3-70B-Instruct + *Stream Aligner*-8B on math, compared to the zero-shot baseline. It is demonstrated that *Stream Aligner* achieves excellent performances across all evaluation metrics for every task. Furthermore, the overall performance of *Stream Aligner* tends to increase with the number of correction rounds, converging to a stable value.

performance, with a maximum win rate of over 76.1% in helpfulness, 36.0% in harmlessness, and 19.0% in math tasks. It's also worth noting that *Stream Aligner*-(2B,8B) model can be utilized to correct the answer of up to 70B models, demonstrating the scalability and efficiency of our method.

Performance on Helpful & Harmless QA Figure 4 shows the distribution shift of helpful and harmless scores after different rounds of sentence-level correction. In terms of helpfulness and harmlessness, due to the instability of the evaluation methods, the performance of *Stream Aligner* shows significant fluctuations, yet there is an overall upward trend in win rates during the early rounds. Beyond a certain number of rounds, the helpfulness of *Stream Aligner* begins to decline, while harmlessness continues to rise. This is because the responses are becoming excessively verbose over time; however, since *Stream Aligner* extensively learns from human preferences, its output remains consistently safe, thus showing a generally upward trend in safety.

Performance on Math Task In math tasks, the performance of *Stream Aligner* monotonically increases with the number of rounds, indicating that *Stream Aligner* performs well in reasoning tasks such as math. In these tasks, we utilized two combinations of *Stream Aligner* models and upstream models: Llama3-8B-Instruct + *Stream Aligner*-2B and Llama3-70B-Instruct + *Stream Aligner*-8B. By training on the suffix preference dataset, both sets of models are able to enhance the math capabilities of the upstream model after certain rounds of correction, as is shown in Figure 3.

Ablation Study

Ablation on *Stream Aligner* Pipeline To verify the correction capabilities of the *Stream Aligner* paradigm with different supervision quantities and different generation pipelines, we conducted ablation studies on the generation methods within the *Stream Aligner* pipeline across all tasks.

- **Generation-Correction Frequency** As seen in Figure 5, the performance of the *Stream Aligner* pipeline increases significantly with the number of generation-correction cycles and continues to rise after surpassing the original model. This demonstrates that *Stream Aligner* can enhance the performance of the upstream model with limited supervision and achieve even higher capabilities under conditions of ample supervision.
- **Generation Methods** We performed an ablation study between two methods within our generation-correction pipeline. The main method, as previously described, iterates through cycles of generation and correction, whereas another method use *Stream Aligner* to continue generation until the end of the answer after the final correction cycle. The performance of these two pipelines on math tasks is shown in Figure 5. This new pipeline demonstrated excellent performance. It reached 80% of its maximum win rate on helpfulness at *1* round and achieved a win rate above zero on math at *1* round, which is significantly lower than the classic pipeline. We hypothesize that this improvement may be attributed to the performance relying heavily on the completeness and correctness of certain key steps rather than distributed across all steps. However, since the continuous generation pipeline is more compute-consuming and converges to a similar result compared to the direct generation pipeline, we still use the direct generation pipeline as the main method.

Ablation on the size of *Stream Aligner* To validate that *Stream Aligner* can fully elicit knowledge of the upstream model, we conducted the ablation study about model sizes. Specifically, we performed the experiment on math task but with different sizes of the upstream model:

- We trained a *Stream Aligner* based on Llama3-70B-Instruct on math tasks. The results indicate that the *Stream Aligner*-70B, after 15 rounds of correction, enhanced the

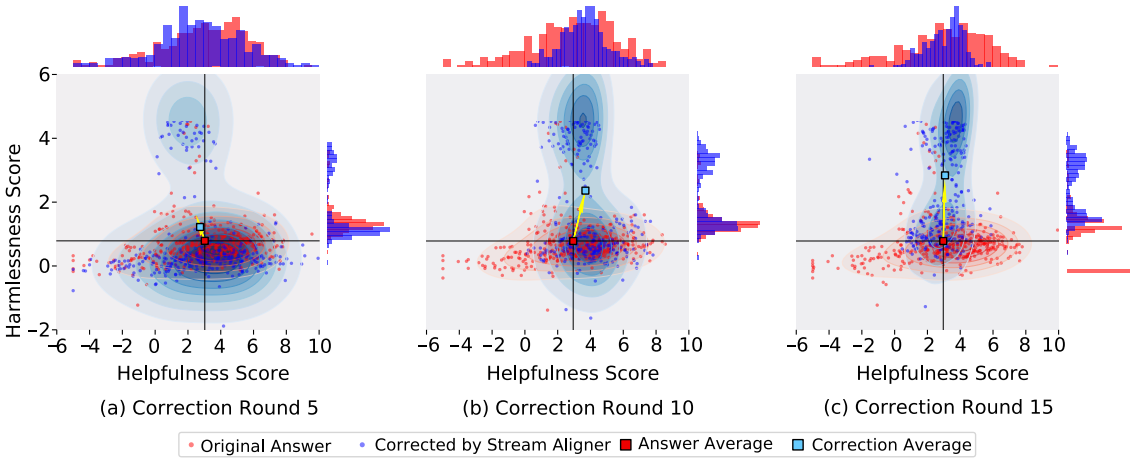


Figure 4: **Distribution of helpful and harmless scores across different rounds. (a-c)** The distribution of helpful and harmless scores during the evaluation generation process. From the development of the distribution, we can find: **(1)** In the first few rounds, since the output length is small, the helpfulness score is relatively low. In the middle and last rounds, the answers can be positively corrected to be helpful **(2)** The harmlessness score of each round continues to increase during the correction generation process. This might because the corrected suffix of *Stream Aligner* is generally safe, and the more corrected output is generated, the higher the safety score would be. **(3)** Overall, we can see the *Stream Aligner* tends to correct the original answer into another distribution, which is more helpful and harmless than the original distribution.

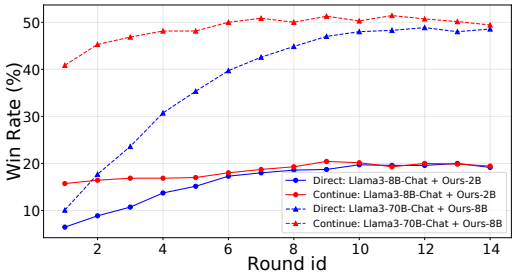


Figure 5: **Ablation of generation pipeline.** We performed an ablation study about different generation pipelines on math tasks and different sizes of upstream models. We observe that under different upstream and *Stream Aligner* models, the new continue generation pipeline exceeds the classical sentence-by-sentence correction pipeline in terms of performance on math tasks. When correction rounds increase, the win rate of both pipelines will eventually converge to a constant value.

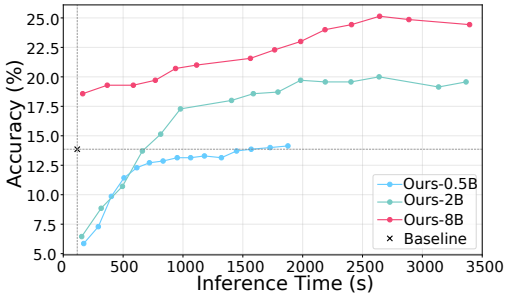


Figure 6: **Ablation on the inference time scaling.** We conducted an ablation study to examine the impact of different additional model sizes on performance across varying inference times, using the same upstream model. Our findings reveal that while larger *Stream Aligner* models achieve superior performance, smaller *Stream Aligner* models also contribute to significant performance improvements for the upstream model. Notably, smaller *Stream Aligner* models reach performance convergence with a shorter inference time, demonstrating their efficiency.

accuracy of Llama3-70B-Instruct by approximately 3.6%, nearly identical to that of the *Stream Aligner*-8B. In the subsequent five rounds, the accuracy improvement by *Stream Aligner*-70B could reach 4.1%.

- We also performed the same experiment on the Llama3-8B-Instruct model, where we compared the inference time and the performance of *Stream Aligner*-8B, *Stream Aligner*-2B, and *Stream Aligner*-0.5B (shown in Figure 6). The result shows that *Stream Aligner*-8B can improve 11.3% accuracy on Llama3-8B-Instruct, but *Stream Aligner*-2B also reached 6.1% improvement using nearly half of the inference time. Moreover, the *Stream Aligner*-0.5B can also improve the performance of Llama3-8B-Instruct, given the huge gap of additional model size.

Considering the huge gap in performance between the models of difference size (Hoffmann et al. 2022) and the narrow gap of accuracy improvement, *Stream Aligner* paradigm can elicit the knowledge of the upstream Model to a large extent (Christiano, Xu, and Cotra 2021).

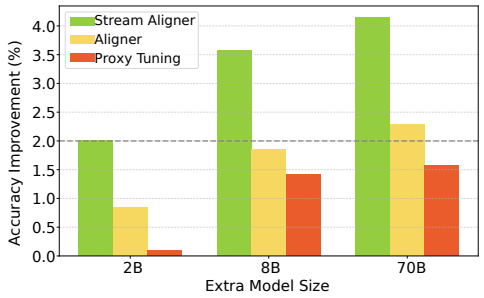


Figure 7: **Ablation of inference time methods.** We conducted an ablation study on math tasks to evaluate different inference-time methods using the same upstream model. The results show that *Stream Aligner* achieves the highest performance improvement across all additional model sizes. Notably, *Stream Aligner* requires only a 2B additional model to achieve performance comparable to a 70B additional model of *Aligner*. This finding highlights that *Stream Aligner* not only achieves a higher performance upper bound but is also significantly more efficient compared to similar methods.

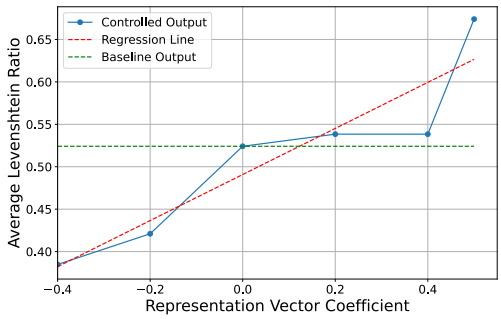


Figure 8: **Representation Control on *Stream Aligner* models.** The control experiment shows the effectiveness of the extracted correction representation vector in modulating the *Stream Aligner*'s correction behavior. The relationship between the Average Levenshtein Ratio and representation vector coefficients is approximately linear, further proving the effectiveness of the representation.

Comparison to other alignment methods To demonstrate the performance enhancements of *Stream Aligner* compared to other alignment methods, we constructed answer preference datasets and sentence-level preference datasets using the same prompt dataset across QA and reasoning tasks. Using these datasets, we conducted an ablation study comparing *Stream Aligner* with other alignment methods, such as SFT and DPO. On Llama3-8B-Instruct, the accuracy improvements for SFT and DPO were -0.5% and 0.3%, respectively, whereas *Stream Aligner* achieved a significant accuracy improvement of 5.8%. These results highlight the superiority of *Stream Aligner* over conventional alignment methods.

We also conducted ablation studies on *Stream Aligner*, *Aligner*, and proxy-tuning (Liu et al. 2024). We reproduced *Aligner* and proxy-tuning on the math task, using the Llama3-70B-Instruct as the upstream model and Llama3.2-2B-Instruct, Llama3-8B-Instruct, Llama3-70B-Instruct as the additional model. The accuracy improvements of *Stream Aligner* and other inference-time methods are presented in Figure 7. Our results demonstrate that *Stream Aligner* consistently outperforms other inference-time methods across all model sizes. Notably, *Stream Aligner* achieves the performance of *Aligner*-70B using only 2B parameters, showcasing both superior performance and efficiency in model size.

Additionally, our experiments reveal that under a standard pipeline generation, the per-token inference time of *Stream Aligner* is only *0.80* times that of *Aligner* when using the same upstream and additional model of the same size. Moreover, the first-token latency of *Aligner* is *10* times higher than that of *Stream Aligner*, demonstrating *Stream Aligner*'s suitability for deployment scenarios.

4 Interpretability of the *Stream Aligner*

Similar to *Aligner*, we observed the correction mechanism in *Stream Aligner*: the correction behavior is not binary between correct and copy, but rather a conditional paradigm and the degree of reference to the original suffix and the extent of extra correction mostly depends on the quality of the original suffix. To demonstrate that *Stream Aligner* learned this correction mechanism as a representation, we conduct the experiment based on *representation engineering* (Zou et al. 2023) and *activation steering* (Turner et al. 2023; Li et al. 2024). Specifically, we perform representation extraction and *Linear Artificial Tomography* (LAT) scan to the *Stream Aligner* model trained for math tasks. We then utilized the extracted representation to control the *Stream Aligner*'s generation.

The ratio of adding (or subtracting) the representation vector in the *Stream Aligner* activation will affect the quantity of correction performed to the original suffix, ranging from directly copying the original response to substantially increasing the extent of normal correction (shown in Figure 8). This provides strong evidence that *Stream Aligner* has internalized the correction paradigm as a representation, just as the *Aligner*. After confirming the effectiveness of the representations, we can analyze the relationship between the representations and mechanisms as mentioned in Figure 9:

- **The similarity of correction mechanism across tasks** The correction mechanism of *Stream Aligner* is quite similar to *Aligner* in terms of representations, where both first decide on the extent of additional correction to introduce and then implement this decision in the remaining layers. This aligns with our initial intent when designing the correction module: building an implicit discriminator and generator within the model.
- **The difference between helpful & harmless QA and math correction tasks** The number of layers involved in deciding correction quantity in *Stream Aligner* seems to be significantly more than in *Aligner*, and similarly, *Stream Aligner* slightly exceeds *Aligner* in the decision-making for copying. This not only confirms the complexity of math tasks compared to helpful & harmless QA but also aligns with the intuitive approach to math tasks: in mathematics, identifying the exact location of an error is usually

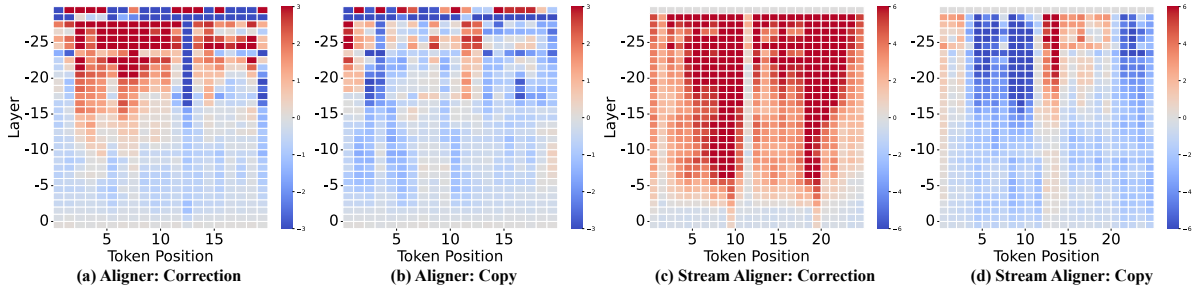


Figure 9: **Interpretability experiment results on *Aligner* and *Stream Aligner***: **(a)(b)** The LAT scan graph of *Aligner*'s each layer. A higher value in the graph indicates a more active correction representation in that layer. **(c)(d)** The LAT scan graph of *Stream Aligner*'s each layer. From the comparison of the LAT scan graphs of *Stream Aligner* and *Aligner*, we can observe that **(i)** *Stream Aligner* has a correction generation mechanism similar to *Aligner*, where the extent of the correction is decided first, followed by the execution of this decision in subsequent layers; **(ii)** there are slight differences in detail between *Stream Aligner* and *Aligner*: the layers where *Stream Aligner* decides to copy are similar to *Aligner*, but *Stream Aligner* has more layers where corrections are decided compared to *Aligner*, which aligns with our intuition about math problems: typically in math problems, once we identify the errors, we can correct them relatively easily.

the main task in correcting it.

5 Related Work

Inference strategies refinement These works aim to refine the inference strategies to acquire better performance without additional training of original models (Chen et al. 2023; Xu et al. 2024; Lu et al. 2023), providing lightweight yet quite effective alignment methods after training. For example, IPA (Lu et al. 2023) incorporates a lightweight policy to replace the calculation of the next-token probability and thereby optimize the performance of the generation, but it directly needs the logit distribution of upstream models, and Speculative sampling (Chen et al. 2023) focuses more on accelerating the generation, rather than performing alignment at inference time.

In our work, the *Stream Aligner* is incorporated into the generation pipeline of upstream models, but it does not directly require the logits and the weights of upstream models, instead, it induces the policy of upstream models and can be applied to various upstream models with only once training.

Incorporating additional model to the inference pipeline These works aim to distill the alignment strategies into an additional model that is incorporated into the inference pipeline without accessing the internal parameters of the upstream models (Ji et al. 2024a; Welleck et al. 2022; Yang and Klein 2021; Dathathri et al. 2019). For example:

- Self-correction (Welleck et al. 2022) trains a self-corrector using an online sampling process, and uses this corrector to correct the output of upstream models directly.
- RAIN (Li et al. 2023) applies a self-evaluation and correction mechanism to refine the output of the upstream model, thereby achieving self-alignment.
- Proxy-tuning (Liu et al. 2024) is a lightweight decoding-time algorithm that uses a small model (expert) and its untuned version (anti-expert) to guide the predictions of an LLM by applying a logit offset based on the differences between the expert and anti-expert outputs.

Compared to previous works, *Stream Aligner* has the following strengths:

- *Stream Aligner* focuses more on eliciting the latent knowledge of the upstream model. In fact, *Stream Aligner* utilizes sentence-level distribution induction rather than directly performing correction on the upstream model output. This elicited the knowledge of upstream models, thus less reliant on the capability of *Stream Aligner* models.
- *Stream Aligner* enable smaller additional models and balance between training efficiency and inference effectiveness, making alignment methods more lightweight.

6 Conclusion

We introduce the Streaming Distribution Induce Aligner (*Stream Aligner*), a novel alignment paradigm that better elicits the latent knowledge of the upstream model and combines efficiency with enhanced performance in various tasks throughout the generation process. *Stream Aligner* has achieved good results in both model size and performance. In helpful & harmless QA, the *Stream Aligner*-2B has managed to improve the helpfulness of the Llama2-70B-Chat model by 41.2%, and harmlessness by 36.0%. Furthermore, the *Stream Aligner*-8B has achieved an improvement of 3.5% in the math ability of the tested Llama3-70B-Instruct model.

Limitations While *Stream Aligner* introduces significant improvements in aligning LLMs through sentence-level dynamic correction, several limitations are left unhandled in this paper, and need further exploration: (1). Although *Stream Aligner* employs smaller models compared to its predecessor, *Aligner*, it still introduces additional computational overhead during the inference phase. (2). *Stream Aligner*'s performance needs relatively high-quality training data, and since *Stream Aligner* uses smaller models, it naturally has trouble dealing with extremely difficult out-of-distribution inputs. (3). Limited to the computation resources, *Stream Aligner* only focuses on two tasks: helpful & harmless QA and math, representing value alignment and knowledge alignment tasks.

Ethical Statement

With its comprehensive composition of preference ranking annotations concerning helpfulness and harmlessness, the *Stream Aligner* model and dataset hold immense potential as a resource for developing beneficial AI assistants aligned with optimal helpfulness and harmlessness along with enhancing the reasoning of LLMs. However, we acknowledge an inherent risk: the same dataset could theoretically be used to train AI assistants in a harmful or malicious manner. As the creators of the *Stream Aligner* model and dataset, we are committed to fostering the development of helpful, safe AI technologies and have no desire to witness any regression of human progress due to the misuse of these technologies. We emphatically condemn any malicious usage of the *Stream Aligner* dataset and advocate for its responsible and ethical use.

Acknowledgements

This work is sponsored by National Natural Science Foundation of China (62376013, 623B2003) and Beijing Municipal Science & Technology Commission (Project ID: Z231100007423015).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Casper, S.; Davies, X.; Shi, C.; Gilbert, T. K.; Scheurer, J.; Rando, J.; Freedman, R.; Korbak, T.; Lindner, D.; Freire, P.; et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*.

Chen, C.; Borgeaud, S.; Irving, G.; Lespiau, J.-B.; Sifre, L.; and Jumper, J. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Christiano, P.; Xu, M.; and Cotra, A. 2021. Arc's first technical report: Eliciting latent knowledge. In *AI Alignment Forum*.

Dai, J.; Pan, X.; Sun, R.; Ji, J.; Xu, X.; Liu, M.; Wang, Y.; and Yang, Y. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; and Liu, R. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.-M.; Chen, W.; et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3): 220–235.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D. d. L.; Hendricks, L. A.; Welbl, J.; Clark, A.; et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hubinger, E.; Denison, C.; Mu, J.; Lambert, M.; Tong, M.; MacDiarmid, M.; Lanham, T.; Ziegler, D. M.; Maxwell, T.; Cheng, N.; et al. 2024. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*.

Ji, J.; Chen, B.; Lou, H.; Hong, D.; Zhang, B.; Pan, X.; Dai, J.; and Yang, Y. 2024a. Aligner: Achieving efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*.

Ji, J.; Liu, M.; Dai, J.; Pan, X.; Zhang, C.; Bian, C.; Chen, B.; Sun, R.; Wang, Y.; and Yang, Y. 2024b. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.

Ji, J.; Qiu, T.; Chen, B.; Zhang, B.; Lou, H.; Wang, K.; Duan, Y.; He, Z.; Zhou, J.; Zhang, Z.; et al. 2023. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*.

Ji, J.; Wang, K.; Qiu, T.; Chen, B.; Zhou, J.; Li, C.; Lou, H.; and Yang, Y. 2024c. Language Models Resist Alignment. arXiv:2406.06144.

Ji, J.; Zhou, J.; Lou, H.; Chen, B.; Hong, D.; Wang, X.; Chen, W.; Wang, K.; Pan, R.; Li, J.; et al. 2024d. Align Anything: Training All-Modality Models to Follow Instructions with Language Feedback. *arXiv preprint arXiv:2412.15838*.

Li, K.; Patel, O.; Viégas, F.; Pfister, H.; and Wattenberg, M. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Li, Y.; Wei, F.; Zhao, J.; Zhang, C.; and Zhang, H. 2023. Rain: Your language models can align themselves without finetuning. *arXiv preprint arXiv:2309.07124*.

Liu, A.; Han, X.; Wang, Y.; Tsvetkov, Y.; Choi, Y.; and Smith, N. A. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Lu, X.; Brahman, F.; West, P.; Jung, J.; Chandu, K.; Ravichander, A.; Ammanabrolu, P.; Jiang, L.; Ramnath, S.; Dziri, N.; et al. 2023. Inference-time policy adapters (ipa): Tailoring extreme-scale lms without fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 6863–6883.

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

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al.

2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Paul, C.; Ajeya, C.; and Xu, M. 2024. Eliciting latent knowledge: How to tell if your eyes deceive you. Google Docs.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 3(6): 7.

Team, G.; Mesnard, T.; Hardin, C.; Dadashi, R.; Bhupatiraju, S.; Pathak, S.; Sifre, L.; Rivière, M.; Kale, M. S.; Love, J.; et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Team, Q. 2024. Introducing Qwen1.5.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Turner, A.; Thiergart, L.; Udell, D.; Leech, G.; Mini, U.; and MacDiarmid, M. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Welleck, S.; Lu, X.; West, P.; Brahman, F.; Shen, T.; Khashabi, D.; and Choi, Y. 2022. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*.

Xu, Z.; Jiang, F.; Niu, L.; Jia, J.; Lin, B. Y.; and Poovendran, R. 2024. SafeDecoding: Defending against Jailbreak Attacks via Safety-Aware Decoding. *arXiv preprint arXiv:2402.08983*.

Yang, K.; and Klein, D. 2021. FUDGE: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.

Yang, K.; Liu, Z.; Xie, Q.; Zhang, T.; Song, N.; Huang, J.; Kuang, Z.; and Ananiadou, S. 2024. MetaAligner: Conditional Weak-to-Strong Correction for Generalizable Multi-Objective Alignment of Language Models. *arXiv preprint arXiv:2403.17141*.

Zou, A.; Phan, L.; Chen, S.; Campbell, J.; Guo, P.; Ren, R.; Pan, A.; Yin, X.; Mazeika, M.; Dombrowski, A.-K.; et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.