
GOOD: Decoding-Time Black-Box LLM Alignment

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Large Language Models (LLMs) have demonstrated immense potential across
2 various applications. However, aligning these models with specific real-world tasks
3 and human preferences typically requires resource-intensive fine-tuning processes
4 such as Supervised Fine-Tuning (SFT) and Reinforcement Learning from Human
5 Feedback (RLHF). In this paper, we propose GOOD (Guided Online Optimal
6 Decoding), a novel alignment method that enhances pre-trained models at decoding
7 time without requiring access to their parameters or vocabularies. We observed that
8 different aligned models exhibit similarities in their decisions of alignment-related
9 tokens. Inspired by this, GOOD utilizes a pair of guiding models to identify critical
10 positions related to alignment and adjusts the model’s output dynamically during
11 the decoding phase. Notably, the interaction between the guiding models and the
12 guided model occurs at the string level, enabling GOOD to be applied to align
13 even black-box models with different vocabularies. Experiments show that in
14 weak-to-strong alignment, GOOD can achieve performance comparable to direct
15 fine-tuning in terms of comprehensive capability and harmless generation, reaching
16 relative scores up to 102% and 99% without sacrificing decoding efficiency. Even
17 when guiding across model families, it can achieve 98% and 103% of the target
18 performance on the two tasks, respectively. Moreover, GOOD can be applied to
19 enhance already aligned models (improving pass@1 by 52% in code enhancement),
20 making it compatible with various existing alignment techniques.

1 Introduction

22 Large Language Models (LLMs) have demon-
23 strated remarkable potential across various ap-
24 plications, such as programming, writing, lan-
25 guage translation, etc. After pre-training on a
26 huge amount of text corpus, they often require
27 further alignment to adapt to specific real-world
28 tasks as well as human values and preferences.
29 The alignment process usually involves Instruc-
30 tion Tuning (Wei et al., 2021) and Preference
31 Learning (Ouyang et al., 2022), typically imple-
32 mented through methods such as SFT (Wei et al., 2021) and RLHF (Ouyang et al., 2022). These
33 alignment methods have significantly enhanced the capabilities of LLMs, suggesting that alignment-
34 related tuning is crucial for developing AI assistants (Bubeck et al., 2023).

35 However, fine-tuning-based alignment methods have three major problems. (1) They are resource-
36 intensive, requiring extensive training data and substantial computational power. (2) The same
37 fine-tuning process is often applied to different models, leading to redundancy. (3) They necessitate

Table 1: Comparison of GOOD and other representative tuning-free alignment methods (Note: “●” indicates No, “●” indicates Yes).

Method	Need extra training?	Need special designed prompt?	Need accessing vocabulary and logits of target model?	Need extra test-time computation?
URIAL	●	●	●	●
RAIN	●	●	●	●
GenARM	●	●	●	●
Proxy-tuning	●	●	●	●
Aligner	●	●	●	●
GOOD	●	●	●	●

38 direct access to the model’s parameters, which is often impractical for state-of-the-art closed-source
39 models (e.g., GPT-4o (OpenAI, 2024)).

40 Given these challenges, there is a growing interest in alignment methods that do not require fine-
41 tuning. Zhou et al. (2024) proposed the *Superficial Alignment Hypothesis*, suggesting that most of
42 a model’s knowledge and capabilities are acquired during pre-training, with alignment primarily
43 teaching the model which sub-distribution of responses to utilize in user interactions. Building
44 on this premise, recent work such as URIAL (Lin et al., 2023) has analyzed token shifts between
45 pre-trained LLMs and their aligned counterparts, finding that most token distribution changes occur
46 in language style-related tokens (e.g., discourse markers, safety disclaimers). RAIN (Li et al., 2023)
47 attempts to use the pre-trained LLMs to evaluate their own generation and use the evaluation results to
48 guide reward and generation for AI safety. Liu et al. (2024) proposed Proxy-Tuning, which achieves
49 an alignment effect similar to direct fine-tuning by computing the logits difference between the
50 pre-trained model and its aligned version, then applying this vector to the output logits of another
51 model in the same model series.

52 Nevertheless, these tuning-free alignment methods face several limitations that restrict their appli-
53 cation across diverse scenarios. (1) Specifically designed in-context prompts cannot fully reflect
54 user preferences across different scenarios, hence failing to adapt flexibly to various types of task.
55 (2) Methods that adjust decoding based on token logits are constrained by the model’s vocabulary,
56 limiting their use to within the same model series. (3) Additionally, existing methods typically incur
57 additional test-time computational costs, rendering them less economically viable. These challenges
58 significantly hinder the practical utility of current non-tuning alignment methods, emphasizing the
59 need for more adaptable and efficient solutions.

60 To this end, we propose **GOOD** (Guided Online Optimal Decoding), a novel tuning-free alignment
61 method. We observed that different aligned models exhibit similarities in their decisions of alignment-
62 related tokens, and alignment-related changes in one model can be used to guide another model (see
63 Appendix A). Based on this observation, we propose the GOOD method, which enhances the model
64 by dynamically adjusting its output during the decoding phase. Specifically, GOOD uses a pair of
65 guiding models to identify critical locations that need alignment during the response generation, and
66 provide corresponding guidance. This identification process is accomplished through a two-step *guess-*
67 *and-verify* mechanism, which probabilistically generates multiple tokens in a single step, thereby
68 achieving a lossless acceleration compared to vanilla decoding. Through this dynamic adjustment,
69 GOOD achieves comparable performance to direct fine-tuning and exhibits high flexibility, making it
70 effective for aligning the behavior of black-box models, where the parameters and vocabulary are
71 not accessible. In Appendix B, we further provide a demonstration of how GOOD is compatible
72 with API-based closed-source model services, requiring only string-level communication between
73 the involved components. Table 1 presents a comparison between GOOD and existing tuning-free
74 alignment methods.

75 Experiments show that in weak-to-strong alignment, GOOD can achieve performance comparable to
76 direct fine-tuning in terms of comprehensive capability and harmless generation, reaching relative
77 scores of 102% and 99%. Meanwhile, it delivers a 3%–13% speedup in decoding time compared with
78 vanilla decoding, achieved through integration with speculative decoding. Even when using guiding
79 models from different model families (often differing in vocabulary, training data, and architecture),
80 GOOD remains effective, achieving 98% and 103% of the target performance on the two tasks,
81 respectively. GOOD can also be applied to enhance already aligned models. In our experiments, the
82 code enhancement from GOOD yielded a 52% relative improvement in the guided model’s pass@1
83 performance. Based on these results, our analysis reveals that the performance improvement brought
84 by GOOD mainly stems from accurately identifying positions that need alignment, and this can be
85 further enhanced by providing more accurate and stronger guidance, suggesting a potential direction
86 for non-tuning alignment to replace tuning-based alignment.

87 We conclude our contributions as follows:

- 88 • To the best of our knowledge, GOOD is the first method to achieve black-box LLM align-
89 ment at decoding time. Distinct from existing tuning-free approaches, GOOD eliminates
90 dependencies on pre-designed contexts and vocabulary constraints while achieving faster
91 decoding than vanilla sampling, combining high flexibility with practical efficiency.
- 92 • We observe that aligned models exhibit consistent patterns in identifying alignment-critical
93 tokens, and model interactions naturally occur at the string level. Building on this insight,

GOOD utilizes a pair of guiding models to implement efficient decoding-time alignment through position-aware guidance that integrates seamlessly with speculative decoding, achieving both alignment effectiveness and decoding efficiency.

- We conducted extensive evaluations across several scenarios. Results show that in weak-to-strong alignment scenarios, GOOD achieves 102% performance of directly fine-tuned models. It also attains 103% relative safety score even when aligned across different model families. Moreover, GOOD successfully enhances already-aligned models, improving pass@1 by 52% in code generation tasks, demonstrating compatibility with existing alignment techniques. These demonstrations broaden the application scope of GOOD.

2 Related Work

2.1 Tuning-based Alignment Methods

Alignment related tuning is critical in adapting LLMs to better reflect human preferences (Wei et al., 2021; Ouyang et al., 2022; Taori et al., 2023; Wang et al., 2023; Rafailov et al., 2024; Bubeck et al., 2023). A common starting point is SFT (Supervised Fine-Tuning), where the model is fine-tuned on datasets containing desired human-instructed outcomes, providing a basic level of alignment. RLHF (Reinforcement Learning from Human Feedback) builds on SFT by incorporating a reward model that guides the policy model towards human-preferred behaviors. There are also several RLHF variants, such as RLAIF (RL from AI Feedback) (Lee et al., 2023), DPO (Direct Preference Optimization) (Rafailov et al., 2024), etc., have been proposed, each aiming to improve the efficiency and effectiveness of the alignment process (Wang et al., 2024). However, these tuning-based methods require considerable resources, including large amounts of training data and significant computational capabilities. Additionally, they require direct access of the model’s parameters, which is often unfeasible for cutting-edge models like GPT-4 (Achiam et al., 2023). In sight of this, some researchers have explored aligning model responses without parameter tuning.

2.2 Tuning-free Alignment Methods

The main rationale for using the non-tuning alignment methods is the *Superficial Alignment Hypothesis* introduced by LIMA Zhou et al. (2024), suggesting that most of a model’s knowledge and capabilities are acquired during pre-training, with alignment primarily teaching the model which sub-distribution of responses to utilize in user interactions. Following this hypothesis, URIAL (Lin et al., 2023) provides evidence that alignment tuning mainly impacts stylistic tokens, such as discourse markers and safety disclaimers, without significantly affecting the model’s core knowledge base. Building on recent advancements in non-tuning alignment research, we categorize related methods into the following three classes.

Pre-decoding alignment methods. URIAL (Lin et al., 2023) leverages In-Context Learning (ICL) (Mann et al., 2020)—a paradigm that enables LLMs to adapt to new tasks through contextual prompts without parameter updates—to achieve pre-decoding alignment. By incorporating few-shot examples (e.g., stylistic demonstrations or inference traces) into prompts, ICL allows LLMs to better align their outputs with user instructions. URAL demonstrates that this approach can attain effective alignment using minimal resources: a system prompt and as few as three constant stylistic examples. Yet, this kind of methods are highly dependent on the design of the few-shot examples, which limits their generalizability and effectiveness in different tasks.

In-decoding alignment methods. In-decoding alignment methods perform adjustments during the model’s response generation, typically achieved by modifying token logits or employing discrimination and search mechanisms. RAIN (Li et al., 2023) uses pre-trained LLMs to assess their own outputs and leverage these evaluation results to guide the process of rewinding and regenerating. Works such as GenARM (Xu et al., 2024), Args (Khanov et al., 2024), Transfer Q-star (Chakraborty et al., 2024), and Cascade Reward Sampling (Li et al., 2024) explore reward-guided decoding from different perspectives. Alternatively, Proxy-tuning (Liu et al., 2024) and EFT (Mitchell et al., 2023) guide generation by injecting logit differences from aligned reference models into target predictions. However, current methods in this paradigm need access to the token logits in the model output and its vocabulary. These factors limits their applicability.

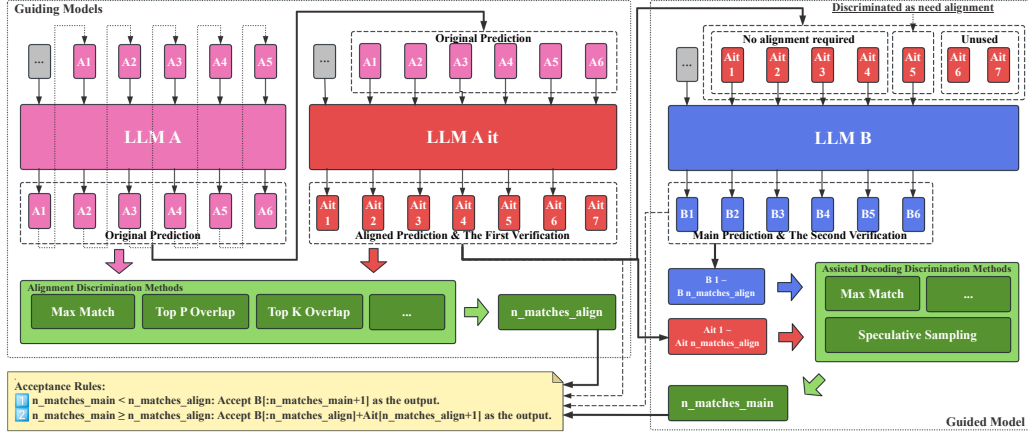


Figure 1: The principle of GOOD. GOOD utilizes a pair of guiding models to identify critical positions related to alignment. Once a specific position is discriminated as requiring alignment, we replace the prediction with the guiding model’s output, converting it to the guided model’s token if needed. LLM_A first predicts multiple tokens, which are then verified by $LLM_{A_{it}}$ (the aligned version of LLM_A). The output from $LLM_{A_{it}}$ is subsequently validated by LLM_B (the guided model). The original version of GOOD without speculative execution, as well as how speculative decoding within GOOD is handled in different scenarios, are provided in Appendix C and Appendix D.

145 **Post-decoding alignment methods.** Aligner (Ji et al., 2024) establishes post-decoding alignment
 146 through a two-stage progress: generating the initial response in the first stage and refining it in the
 147 second stage. It trains a separated model that learns correctional residuals between initial and aligned
 148 outputs without the need for fine-tuning the base LLM. Nevertheless, the effectiveness of Aligner is
 149 limited by the initial generation step, which makes it difficult to align responses if the base model
 150 produces poor answers. Additionally, it still requires fine-tuning of the downstream model.

151 2.3 LLM Ensemble

152 LLM ensemble methods leverage multiple models, each contributing unique insights and diverse
 153 reasoning patterns, thereby compensating for individual model weaknesses and reducing biases. Lu
 154 et al. (2024) provides a more detailed introduction. Taking the GaC method (Yu et al., 2024) as an
 155 example, GaC treats each token generation as a classification task and averages the classification
 156 probability vectors across multiple LLMs during inference. This approach utilizes the token-level
 157 probability information from each model and integrates multiple models at the inference stage,
 158 improving overall performance and preventing early-generation errors from cascading into larger
 159 mistakes.

160 2.4 Speculative Decoding

161 Recent work on speculative decoding has shown that large autoregressive language models can be
 162 decoded significantly faster by combining a fast “draft” model with the original, more powerful
 163 “target” model (Leviathan et al., 2023; Xia et al., 2022; Chen et al., 2023; Miao et al., 2024). This
 164 approach generates several candidate tokens in parallel from smaller or more efficient models (the
 165 draft model), then relies on the larger (target) model to validate these tokens in a single verification
 166 step. GOOD integrates the concept of speculative decoding, combining alignment discrimination
 167 and token generation into a dual-stage speculation-verification process, achieving both non-tuning
 168 alignment and acceleration of target model decoding.

169 3 Method

170 In this section, we introduce the principles of GOOD (Guided Online Optimal Decoding), with
 171 an overview provided in Figure 1. The original version of GOOD without speculative execution
 172 is provided in Appendix C, clearly demonstrating its core principles. The goal of GOOD is to
 173 achieve flexible and efficient tuning-free alignment, without accessing the parameters, logits, or
 174 vocabulary of the target model. Appendix B provides a demonstration of how GOOD is compatible

with API-based closed-source model services, requiring only string-level communication between the involved components.

We first formalize the problem setting and notation, then detail the two key components of GOOD: (1) discriminating which positions need alignment, and (2) the transformation of guidance (including token conversion across vocabularies and alignment flag updates). Finally, we present the overall process, incorporating the speculative verification mechanism.

3.1 Problem Setting and Notation

Let $\mathbf{B}$ be the *guided* model that we aim to align, but for which we only have black-box (string-based) access. We assume access to a guiding model $\mathbf{A}$ (the unaligned version), and its aligned variant $\mathbf{A}_{it}$. We denote tokenizers as follows: $T_A, T_{A_{it}}$ for the guiding pair, and T_B for the guided model.

A single decoding step at position n generates the next token t^n . We write $p_A(t \mid t^{[1:n-1]})$, $p_{A_{it}}(t \mid t^{[1:n-1]})$, and $p_B(t \mid t^{[1:n-1]})$ for the probability (logit) distribution of the next token, conditioned on the partial sequence $t^{[1:n-1]}$.

We wish to produce an output that is aligned to human preferences (following instructions, safety constraints, etc.), even though $\mathbf{B}$ itself is not aligned. Our approach will replace certain tokens (or sequences of tokens) in $\mathbf{B}$'s raw decoding with corresponding tokens from $\mathbf{A}_{it}$, guided by a token-level alignment discrimination through comparing p_A and $p_{A_{it}}$.

3.2 Alignment Discrimination

We define a function $f(\cdot)$ to decide whether to align at each step: $\delta_n = f(\{p_A(t \mid t^{[1:n-1]})\}, \{p_{A_{it}}(t \mid t^{[1:n-1]})\})$, where $\delta_n \in \{0, 1\}$ is an alignment flag, indicating "no alignment needed" or "alignment needed" at position n .

Here we list two variants of f :

1. **Max-Match:** Compare the single highest-probability token for $\mathbf{A}$ vs. $\mathbf{A}_{it}$. Formally, if $\arg \max_t p_A(t \mid t^{[1:n-1]}) \neq \arg \max_t p_{A_{it}}(t \mid t^{[1:n-1]})$, then $\delta_n = 1$; otherwise 0.
2. **Top- P/K Overlap:** For guiding model $\mathbf{A}$, we define $S_{\text{top}P}^A$ as the minimal set of highest-probability tokens whose cumulative probability exceeds P , and $S_{\text{top}K}^A$ as the top- K highest-probability tokens. Similarly, define $S_{\text{top}P}^{A_{it}}$ and $S_{\text{top}K}^{A_{it}}$ for $\mathbf{A}_{it}$. Then we decide: $\delta_n = 1$ if $|S_{\text{top}P/K}^A \cap S_{\text{top}P/K}^{A_{it}}| < \tau$; otherwise 0, where τ is a threshold that is a nonnegative integer. This approach allows the alignment sensitivity to be easily adjusted by simply adjusting τ .

3.3 Guidance Transformation

Whenever $\delta_n = 1$, we seek to replace $\mathbf{B}$'s next token with the prediction from $\mathbf{A}_{it}$. Considering $\mathbf{A}_{it}$ and $\mathbf{B}$ may have different vocabularies, we process substitutions at the string level to preserve context equivalence. Formally:

Token-to-String: Let $t_{A_{it}}^{\text{new}[1:m]}$ denote the newly predicted m tokens from $\mathbf{A}_{it}$ at the current step. Convert these tokens into a substring: $s_{\text{new}} = T_{A_{it}}^{-1}(t_{A_{it}}^{\text{new}[1:m]})$.

Re-tokenize: Tokenize s_{new} into $\mathbf{B}$'s vocabulary: $t_B^{\text{new}[1:n]} = T_B(s_{\text{new}})$, where n may differ from m due to vocabulary mismatches.

Alignment Flag Update: For each token $t_B^{\text{new}[i]}$ in $\mathbf{B}$'s sequence, identify all tokens $t_{A_{it}}^{\text{new}[j]}$ from $\mathbf{A}_{it}$ that contribute to its formation via string-level mapping, including direct 1-to-1 token mapping, substrings of $t_{A_{it}}^{\text{new}[j]}$, or multi-token overlaps from $\mathbf{A}_{it}$.

Set the alignment flag for $t_B^{\text{new}[i]}$ as: $\delta_B^{\text{new}[i]} = 1$ if $\exists j$ s.t. $t_{A_{it}}^{\text{new}[j]}$ contributes to $t_B^{\text{new}[i]}$ and $\delta_A^{\text{new}[j]} = 1$; otherwise 0. The updated alignment count is then: $n_{\text{matches_align}} = \min\{i \mid \delta_B^{\text{new}[i]} = 1\}$.

3.4 Overall Algorithm

Building upon the alignment discrimination mechanism (3.2) and guidance transformation process (3.3), we present the complete GOOD algorithm through pseudocode in Algorithm 1 (see Appendix E).

Here, we provide a simplified description of the workflow for the GOOD algorithm:

1. Speculative generation with alignment discrimination: Generate draft tokens using the unaligned model (**A**) and validate them with the aligned model (**A**_{it}). Identify positions requiring alignment ($\delta_n = 1$) using a discrimination function.
2. Cross-model guidance transformation: Transform the validated token sequence into the vocabulary of the guided model (**B**) while correspondingly converting the alignment flags (δ).
3. Target model validation: Feed the transformed tokens into the guided model (**B**) for validation. Obtained the final output based on acceptance rules.

4 Experiment

We conducted four experiments to test the capabilities of GOOD: comprehensive performance, harmless generation, enhancing aligned models, and the speed of decoding.

Tasks and datasets. We use MT-Bench (Zheng et al. (2023)) to evaluate the comprehensive performance of GOOD, a multi-task benchmark designed to assess the ability of models across various domains. To evaluate the ability of the GOOD to generate harmless responses, we conducted experiments on the Helpful and Harmless (HH) dataset (Ganguli et al., 2022), designed to test how models perform in complex and sensitive scenarios. In the experiment to enhance the capabilities of already aligned models, we focused on improving code generation skills and evaluated performance on the HumanEval dataset (Chen et al., 2021). In the decoding speed experiments, considering the diversity of tasks, we also used the MT-Bench dataset as the test input.

Models. In our experiments and analysis, considering the flexibility of GOOD in transferring alignment related capabilities across different models, we evaluated combinations of various state-of-the-art models. Specifically, we used the Llama series (Llama-2 (Touvron et al., 2023), CodeLlama (Roziere et al., 2023)), the Gemma series (Gemma-2 (Team et al., 2024)), and Qwen series (Qwen2 (Yang et al., 2024)) to assess the method’s performance and generality.

4.1 Comprehensive Evaluation

On MT-Bench, we tested the effectiveness of weak-to-strong guidance in the Gemma2, Llama2, and Qwen2 series, as well as the cross-family guidance provided by the Gemma2 series to the Qwen2 series models. In the latter case, we used small guiding model pairs from the same series as the alignment discriminator and applied guidance from Gemma2 at positions identified as alignment-related. As shown in the Table 2, whether for guidance within the same series or across different series, GOOD-guided alignment achieved performance comparable to direct fine-tuning. In the case of Llama-2-7b-chat guiding Llama-2-70b, the alignment performance even surpassed that of direct fine-tuning. In comparison with the baseline method (Proxy-Tuning), GOOD outperformed in all three configurations and demonstrated more stable performance (the baseline method did not perform as well on the Gemma2 series).

Table 2: MT-Bench scores for different models and methods.

Method	Model	MT-Bench Score
GOOD	Gemma-2-2b-it → Gemma-2-27b	8.30
	Llama-2-7b-it → Llama-2-70b	6.91
	Qwen-2-7b-it → Qwen-2-72b	8.48
GOOD(Split)	Gemma-2-9b-it + Qwen-2-7b-it → Qwen-2-72b	8.64
Proxy-Tuning	Gemma-2-2b-it → Gemma-2-27b	3.70
	Llama-2-7b-it → Llama-2-70b	6.41
	Qwen-2-7b-it → Qwen-2-72b	8.47
Baseline	Gemma-2-27b-it	8.97
	Llama-2-70b-it	6.86
	Qwen-2-72b-it	9.12

4.2 Harmless Generation

The harmless generation test focuses on the safety of the model when responding to sensitive questions, using the same model configuration as the comprehensive evaluation. We use gpt-4o (Hurst et al., 2024) as the evaluator, the prompt used for evaluation is shown in Appendix H. The harmless ratios for various model settings are summarized in Table 3, demonstrating the improvements achieved through the guiding alignment process.

Under the guidance of smaller models within the same series, we achieved 99% (Gemma2), 98% (Llama2), and 97% (Qwen2) alignment performance relative to direct fine-tuning in the three model configurations. Compared to the baseline method (Proxy-Tuning), GOOD outperformed in two configurations and demonstrated greater stability. Notably, by introducing external guidance across model families, the harmlessness ratio in the GOOD(Split) configuration surpassed that of the directly fine-tuned guided model (74.6% vs 73.0%), highlighting the advantages brought by GOOD’s flexibility.

To further validate GOOD’s effectiveness on harmless generation, we conducted additional comparisons against prominent reward-based decoding-time alignment methods, following the experimental setup of GenARM (Xu et al., 2024). In GenARM, the LLaMA-7B-SFT checkpoint provided by Khanov et al. (2024) is used as the base model, which is fine-tuned from LLaMA-7B on the preferred responses of the HH-RLHF. For both RM and DPO, they fine-tune LLaMA-7B-SFT with LoRA for one epoch on the training split of HH-RLHF. We used TinyLlama-1.1B-Chat and TinyLlama-1.1B as a pair of guiding models (Zhang et al., 2024). And we followed the model preparation process of GenARM to performed the same DPO training on TinyLlama-1.1B-Chat.

Here we report the comparison results in 4. Despite leveraging significantly smaller guiding models, GOOD achieves competitive alignment performance, outperforming several reward-based methods (ARGS, Transfer-Q, CARDS) and approaching the performance of GenARM.

4.3 Enhance Aligned Model

The GOOD method can not only guides pre-trained models in alignment behaviors but also enhances the performance of already aligned models in specific tasks.

Our experiment is evaluated based on the HumanEval dataset. We used CodeLlama-7b-python and Llama-2-7b as the guiding model pair in the GOOD method to enhance the code performance of Llama-2-13b-chat (as the guided model), with $Top_P_A=0.8$ and $Top_P_{A_{it}}=0$. Consistent with Proxy-Tuning (Liu et al., 2024), we set Top_P (sampling parameter) to 0.95, temperature to 0.1, and calculated the pass@1 score. According to the definition provided in Lu et al. (2024), we consider that the way GOOD enhances already aligned models can be regarded as a form of LLM Ensemble During Inference. Therefore, we also compared it with the recently proposed GaC method (Yu et al., 2024).

The detailed performance results are shown in Table 5, where our method achieved a score of 32.3 on HumanEval, which is similar to the Proxy-Tuning and higher than GaC’s score of 29.9. The prompt used in our evaluation is shown in Appendix F. Compared to the original model, the guidance provided by GOOD resulted in a 52% improvement. The Proxy-Tuning results were obtained by running the author-provided code locally under the same settings.

Table 3: Harmless ratios for different models and methods, evaluated by gpt-4o.

Method	Model	Harmless ratio
GOOD	Gemma-2-2b-it → Gemma-2-27b	74.7%
	Llama-2-7b-it → Llama-2-70b	74.7%
	Qwen-2-7b-it → Qwen-2-72b	70.6%
GOOD(Split)	Gemma-2-9b-it + Qwen-2-7b-it → Qwen-2-72b	74.6%
Proxy-Tuning	Gemma-2-2b-it → Gemma-2-27b	54.3%
	Llama-2-7b-it → Llama-2-70b	77.6%
	Qwen-2-7b-it → Qwen-2-72b	68.3%
Baseline	Gemma-2-27b-it	75.6%
	Llama-2-70b-it	76.6%
	Qwen-2-72b-it	73.0%

Table 4: Comparison of GOOD with reward-based decoding-time alignment methods on HH-RLHF dataset.

Method vs. DPO	Win (%)	Tie (%)	Lose (%)	Win + ½ Tie (%)
ARGS	24.44	4.89	70.67	26.89
Transfer-Q	31.00	5.44	63.56	33.72
CARDS	37.89	8.11	54.00	41.94
GenARM	48.00	6.89	45.11	51.44
GOOD (ours)	41.67	5.67	52.67	44.50

Table 5: Pass@1 scores on HumanEval. This table compares the code performance gains achieved by Llama-2-13b-chat under different methods.

Method	HumanEval Pass@1
Llama-2-13b-chat	21.3
CodeLlama-7b-python	38.4
CodeLlama-7b-python + Llama-2-13b-chat (GaC)	29.9
CodeLlama-7b-python → Llama-2-13b-chat (Proxy-Tuning)	32.1
CodeLlama-7b-python → Llama-2-13b-chat (GOOD)	32.3

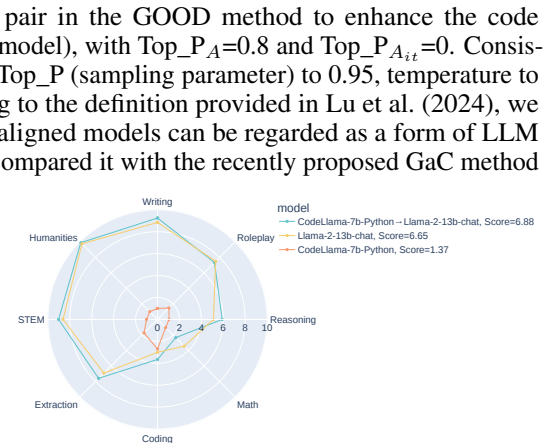


Figure 2: Comprehensive performance with code enhancement guidance, utilized code block markers as the start and end signals.

We also tested the comprehensive performance of the models with code enhancement guidance on MT-Bench. We used the default configuration of GOOD (Max Match) and utilized code block markers as the start and end signals for enhanced guidance (A specific example is shown in Appendix G). As shown in Figure 2, experimental results indicate that models with GOOD-enhanced guidance can surpass both the original and guiding models in comprehensive performance, with score increasing from 6.65 to 6.88.

4.4 Decoding Speed of GOOD

By combining with speculative decoding, GOOD can probabilistically generate multiple tokens in a single iteration or skip the inference steps of the guided model. Considering the diversity of test samples, we used the question set from the MT-Bench dataset as input for decoding speed test (which includes 8 types of tasks). As shown in Figure 3, GOOD outperforms vanilla decoding in decoding speed across all three configurations, achieving up to a 13% speedup. Due to different memory requirements for different model configurations, the tests for Gemma-2-2B-Instruct $\rightarrow$ Gemma-2-27B were conducted on L40s $48G \times 8$, while the other two were tested on A100 $80G \times 8$. The respective baselines were evaluated in the corresponding testing environments.

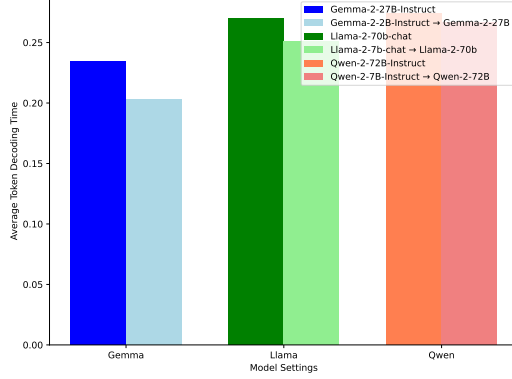


Figure 3: Average token decoding time of GOOD and Vanilla Decoding.

5 Analysis

5.1 Where does the performance enhancement mainly come from?

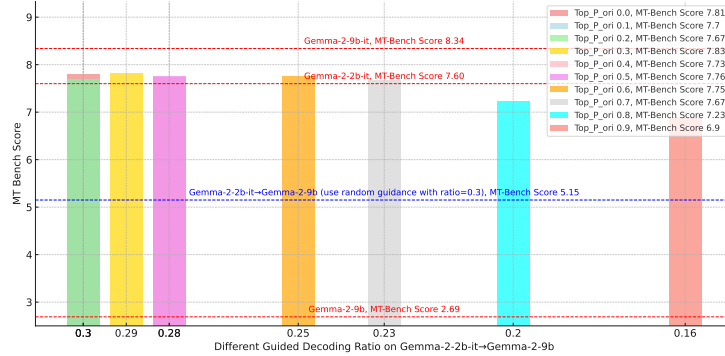


Figure 4: Performance of alignment guidance with varying guided decoding ratios.

To illustrate why the guidance provided by GOOD can help the model achieve performance gains, we evaluated the guided decoding ratio (controlled by adjusting Top_P_A) and MT Bench performance under different parameter configurations, and compared them with random decoding. Based on URIAL’s definition of token shift, we fixed $Top_P_{A_{it}}$ to 0 and adjusted the size of Top_P_A . Due to potential differences in vocabularies between guiding models and the guided model in GOOD, we count the number of guided decodings and original decodings based on the character level in the final results. As shown in Figure 4, the scores of alignment guidance consistently range from 7.67 to 7.83 as the proportion of guided decodings decreases from 0.30 to 0.23.

Even with approximately a 23% reduction in guided decodings (from 0.3 to 0.23), the performance does not experience significant changes. Meanwhile, when random guided decoding at a 0.3 ratio was provided, the model’s performance was significantly lower than that of GOOD-guided decoding. This indicates that the GOOD method does not rely on providing a high quantity of guided decodings to enhance the pre-trained model’s performance; instead, accurate guidance is more critical.

5.2 Token changes in GOOD-guided decoding

To understand the alignment behavior characteristics of models guided by GOOD, we compared the token changes between models aligned using the GOOD method and those aligned directly through fine-tuning, with statistics derived from their responses on the MT-Bench dataset.

We counted the top 100 most frequently changing tokens in each setting. Results show that in the guidance of Llama-3-8B-Instruct to Qwen2-7B, the token changes overlap 70% with Llama-3-8B-Instruct and 64% with Qwen2-7B-Instruct. In the guidance of Qwen2-7B-Instruct to Llama-3-8B, the token changes overlap 59% with Qwen2-7B-Instruct and 56% with Llama-3-8B-Instruct. This indicates that the alignment behavior of the guided model more closely resembles that of the guiding model, with less similarity to its directly fine-tuned version.

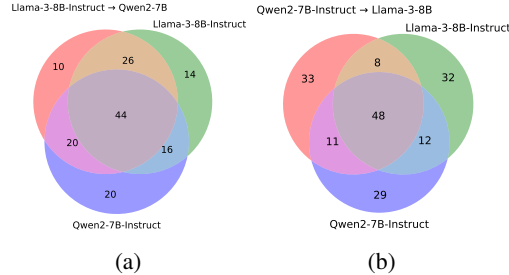


Figure 5: Comparison of token changes in guided decoding alignments.

5.3 More accurate identification as well as stronger guidance.

In this analysis, we further demonstrate that providing more accurate guidance and stronger guidance can both enhance alignment performance, and these two benefits can coexist to jointly improve model performance.

We measured the performance of guiding Qwen2-72B with Gemma-2-9b-it (using both discrimination and guidance from Gemma-2-9b-it) and compared it with the data from Experiment 4.1. Since Qwen2-7B-Instruct and Qwen2-72B belong to the same model family and are trained on the same dataset, Qwen2-7B-Instruct offers more accurate recognition than Gemma-2-9b-it. Meanwhile, Gemma-2-9b-it has a higher score on MT-Bench, indicating it can provide stronger guidance at the same decoding positions. As shown in Figure 6, the results demonstrate that the configuration combining Qwen2-7B-Instruct’s discrimination with Gemma-2-9b-it’s guidance outperforms using Qwen2-7B-Instruct or Gemma-2-9b-it as guidance individually.

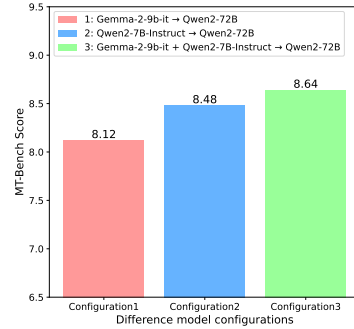


Figure 6: Alignment performance when using more accurate identification as well as stronger guidance.

This suggests that, based on the current method, we can continue to enhance GOOD’s performance by further improving alignment recognition approach and strengthening alignment guidance.

6 Conclusion

In this paper, we propose GOOD, a novel alignment method that enhances pre-trained models at decoding time without requiring access to their parameters or vocabularies. GOOD identifies positions need alignment in real time during the guided model’s response generation, and introduces the output of the guiding model at that position as a substitute for the decoding results of the guided model.

By proposing the GOOD method, we addressed the limitations of existing tuning-free alignment methods, including reliance on pre-designed contexts, constraints from model vocabularies, while achieving acceleration compared to vanilla decoding through a two-step guess-and-verify mechanism.

Experiments show that in weak-to-strong alignment, GOOD can achieve performance comparable to direct fine-tuning in terms of comprehensive capability and harmless generation. Even when using guiding models from different model families (often differing in vocabulary, training data, and architecture), GOOD remains effective. GOOD can also be applied to enhance already aligned models. Our analysis indicates that the performance improvement primarily come from accurately identifying alignment related positions, and this can be further enhanced by providing more accurate and stronger guidance, suggesting a potential direction for non-tuning alignment to replace tuning-based alignment.

406 A discussion of the current limitations of GOOD and potential areas for future work is provided in
407 Appendix I.

408 References

- 409 Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt,
410 J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- 411 Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J. A., Horvitz, E., Kamar, E., Lee, P., Lee,
412 Y. T., Li, Y.-F., Lundberg, S. M., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of
413 artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712, 2023. URL
414 <https://api.semanticscholar.org/CorpusID:257663729>.
- 415 Chakraborty, S., Ghosal, S. S., Yin, M., Manocha, D., Wang, M., Bedi, A. S., and Huang, F. Transfer
416 q-star: Principled decoding for llm alignment. *Advances in Neural Information Processing Systems*,
417 37:101725–101761, 2024.
- 418 Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large
419 language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.
- 420 Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y.,
421 Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint*
422 *arXiv:2107.03374*, 2021.
- 423 Ganguli, D., Lovitt, L., Kernion, J., Askell, A., Bai, Y., Kadavath, S., Mann, B., Perez, E., Schiefer,
424 N., Ndousse, K., et al. Red teaming language models to reduce harms: Methods, scaling behaviors,
425 and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.
- 426 Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A.,
427 Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- 428 Ji, J., Chen, B., Lou, H., Hong, D., Zhang, B., Pan, X., Dai, J., and Yang, Y. Aligner: Achieving
429 efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*, 2024.
- 430 Khanov, M., Burapachee, J., and Li, Y. Args: Alignment as reward-guided search. *arXiv preprint*
431 *arXiv:2402.01694*, 2024.
- 432 Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K. R., Bishop, C., Hall, E., Carbune,
433 V., Rastogi, A., et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai
434 feedback. In *Forty-first International Conference on Machine Learning*, 2023.
- 435 Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding.
436 In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.
- 437 Li, B., Wang, Y., Lochab, A., Grama, A., and Zhang, R. Cascade reward sampling for efficient
438 decoding-time alignment. *arXiv preprint arXiv:2406.16306*, 2024.
- 439 Li, Y., Wei, F., Zhao, J., Zhang, C., and Zhang, H. Rain: Your language models can align themselves
440 without finetuning. *arXiv preprint arXiv:2309.07124*, 2023.
- 441 Lin, B. Y., Ravichander, A., Lu, X., Dziri, N., Sclar, M., Chandu, K., Bhagavatula, C., and Choi, Y.
442 The unlocking spell on base llms: Rethinking alignment via in-context learning. In *The Twelfth*
443 *International Conference on Learning Representations*, 2023.
- 444 Liu, A., Han, X., Wang, Y., Tsvetkov, Y., Choi, Y., and Smith, N. A. Tuning language models by
445 proxy. *arXiv preprint arXiv:2401.08565*, 2024.
- 446 Lu, J., Pang, Z., Xiao, M., Zhu, Y., Xia, R., and Zhang, J. Merge, ensemble, and cooperate! a survey
447 on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089*,
448 2024.
- 449 Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry,
450 G., Askell, A., Agarwal, S., et al. Language models are few-shot learners. *arXiv preprint*
451 *arXiv:2005.14165*, 1, 2020.

452 Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L.,
453 Shi, X., et al. Specinfer: Accelerating large language model serving with tree-based speculative
454 inference and verification. In *Proceedings of the 29th ACM International Conference on Archi-
455 tectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949,
456 2024.

457 Mitchell, E., Rafailov, R., Sharma, A., Finn, C., and Manning, C. D. An emulator for fine-tuning
458 large language models using small language models. *arXiv preprint arXiv:2310.12962*, 2023.

459 OpenAI. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

460 Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S.,
461 Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback.
462 *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

466 Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R.,
467 Remez, T., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*,
468 2023.

469 Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B.
470 Stanford alpaca: An instruction-following llama model, 2023.

471 Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard,
472 T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size.
473 *arXiv preprint arXiv:2408.00118*, 2024.

474 Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S.,
475 Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv
476 preprint arXiv:2307.09288*, 2023.

477 Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan,
478 K., Smith, N. A., Beltagy, I., and Hajishirzi, H. How far can camels go? exploring the state
479 of instruction tuning on open resources. *ArXiv*, abs/2306.04751, 2023. URL [https://api.
480 semanticscholar.org/CorpusID:259108263](https://api.semanticscholar.org/CorpusID:259108263).

481 Wang, Z., Bi, B., Pentyala, S. K., Ramnath, K., Chaudhuri, S., Mehrotra, S., Mao, X.-B., Asur, S.,
482 et al. A comprehensive survey of llm alignment techniques: Rlhf, rlaf, ppo, dpo and more. *arXiv
483 preprint arXiv:2407.16216*, 2024.

484 Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V.
485 Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

486 Xia, H., Ge, T., Chen, S.-Q., Wei, F., and Sui, Z. Speculative decoding: Lossless speedup of
487 autoregressive translation. 2022.

488 Xu, Y., Sehwag, U. M., Koppel, A., Zhu, S., An, B., Huang, F., and Ganesh, S. Genarm: Reward
489 guided generation with autoregressive reward model for test-time alignment. *arXiv preprint
490 arXiv:2410.08193*, 2024.

491 Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al.
492 Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

493 Yu, Y.-C., Kuo, C.-C., Ye, Z., Chang, Y.-C., and Li, Y.-S. Breaking the ceiling of the llm community
494 by treating token generation as a classification for ensembling. *arXiv preprint arXiv:2406.12585*,
495 2024.

496 Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model. *arXiv
497 preprint arXiv:2401.02385*, 2024.

- 498 Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.,
499 et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information*
500 *Processing Systems*, 36:46595–46623, 2023.
- 501 Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., et al. Lima: Less
502 is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

A Observation: Different aligned models exhibit similarities in their decisions of alignment-related tokens

Here we examine the alignment-related token decisions in three models from the Gemma2 series (2b, 9b, and 27b). For each model, the most frequently changed tokens are as follows:

Gemma2-2b-it:

```
['and', 'the', 'a', ',', 'The', '.', '\n\n', '\n', '**', '\n<end_of_turn>', ',',  
'for', '**', ':', 'is', 'in', 'This', '(', 'to', 'with', 'of', '"', 'A', '"', 'that',  
'an', 'We', 'it', 'me', 'how', "Here'", '<end_of_turn>', 'are', 'like', 'I', 'It',  
'this', '-', ':**', '\n\n\nLet', 'potential', 'or', '\n\n*', 'from', 'on', 'can',  
'specific', '!', 'more', 'you', ...]
```

Gemma2-9b-it:

```
['a', 'and', ',', '\n\n', 'the', '**', 'The', '<end_of_turn>', '.', '\n', 'to',  
'**', '(', '(', '(', '"', ':', '-', 'in', 'This', '\n\n*', 'for', 'A', 'are', 'of', 'on',  
'with', 'that', 'is', 'you', 'Here', 'it', "Here'", '"', 'like', 'It', '\n\n**',  
'##', 'I', 'by', '\n\n\n<end_of_turn>', ':**', 'how', '1', 'from', 'potential', 'We',  
'its', 'me', 'if', 'both', ...]
```

Gemma2-27b-it:

```
['a', 'and', ',', '\n\n', 'the', '**', 'The', '<end_of_turn>', '.', '\n', 'to',  
'**', '(', '(', '(', '"', ':', '-', 'in', 'This', '\n\n*', 'for', 'A', 'are', 'of', 'on',  
'with', 'that', 'is', 'you', 'Here', 'it', "Here'", '"', 'like', 'It', '\n\n**', '##',  
'I', 'by', '\n\n\n<end_of_turn>', ':**', 'how', '1', 'from', 'potential', 'We', 'its',  
'me', 'if', 'both', ...]
```

When examining the top 100 most frequent alignment-related tokens, we observed over 70% overlap between the three models. And this overlap increased to over 80% when considering only the top 50 tokens. This observation suggests that there is a considerable similarity in the alignment-related token decisions across different models (especially for those high-frequency tokens). Inspired by this, we explored leveraging alignment decisions from already aligned models to guide pretrained models, and ultimately proposed the GOOD method.

B Compatibility with API-based Closed-Source Models

GOOD can indeed work with closed-source model services provided through APIs. However, this requires some adjustments to the existing API service format. Here, we provide the following explanation and introduce a demonstration we implemented to provide GOOD compatible LLM services in an API format.

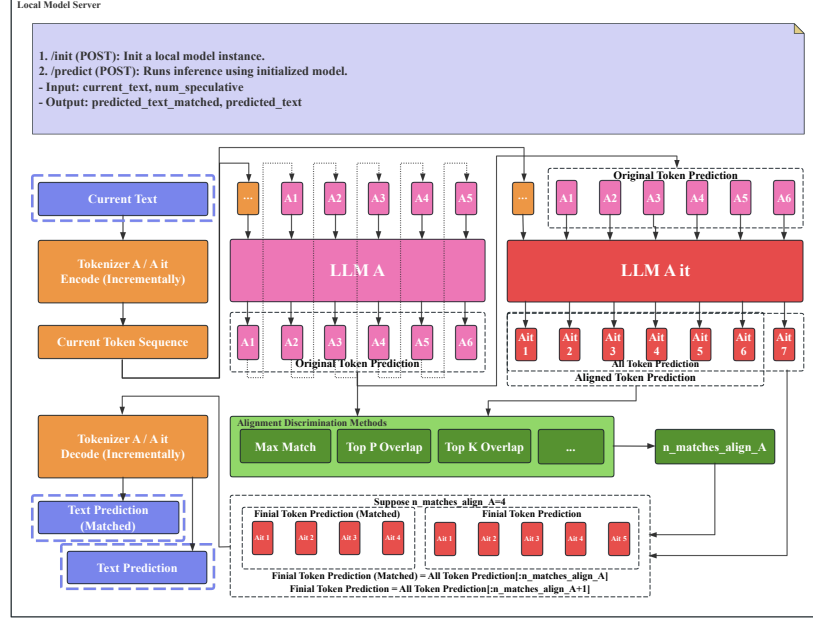
Limitations of Current API Formats for GOOD Integration. Current closed-source model providers typically offer LLM services in the form of APIs, which usually return completed text responses at once or continuously send text fragments in a streaming output format. During a single response generation, adjustments requests from user are not allowed, which makes them not directly compatible with GOOD’s requirements (in GOOD, the guided model needs to continuously receive text information from the guiding model pair to achieve real-time alignment adjustments).

Feasibility and Confidentiality Considerations. However, GOOD can be applied to powerful closed-source models with simple adjustments to the existing API service format. Closed-source model providers hope to avoid leaking the model’s confidential information during providing LLM responses, which is mainly the model’s parameters and vocabulary. During the GOOD process, the guiding models does not need accessing parameters and vocabulary, so its working principle has no fundamentally conflict to the need of protecting model confidentiality.

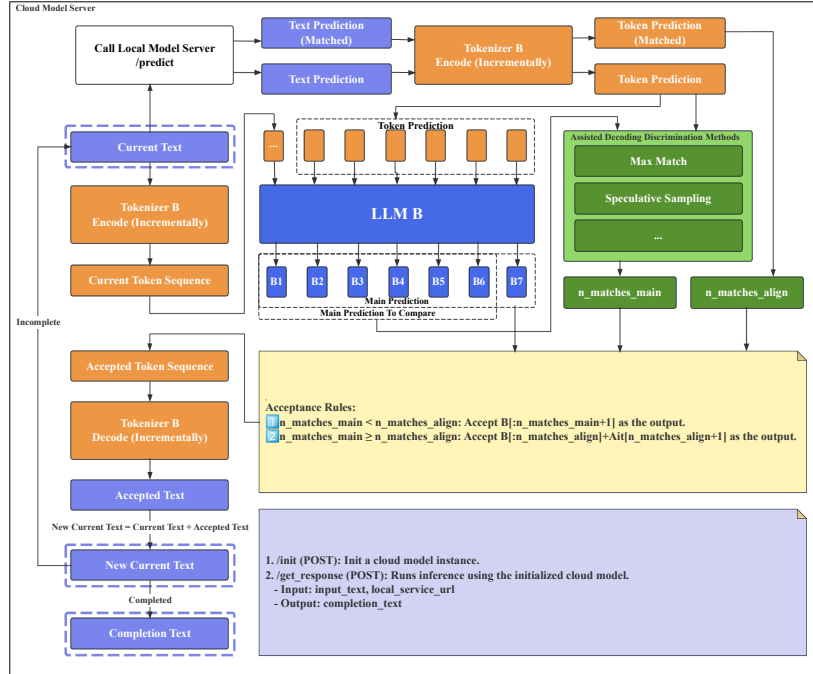
Demonstration of a GOOD-Compatible API Service. Here we provide a demonstration to illustrate the format of the API service required by GOOD, which can be achieved by adjusting the existing API service format. It is worth noting that we have implemented a demo of this service locally and



(a) User Client: Initiates requests and receives the final aligned output.



(b) Local Model Server: Hosts the guiding model pair (LLM_A , $LLM_{A_{lit}}$) and provides alignment guidance as text fragments.



(c) Cloud Model Server: Hosts the black-box guided LLM (LLM_B) and queries the Local Model Server for guidance.

Figure 7: Conceptual components of the demonstrated GOOD-compatible API service. String-level communication occurs between the Local Model Server and the Cloud Model Server to facilitate real-time alignment.

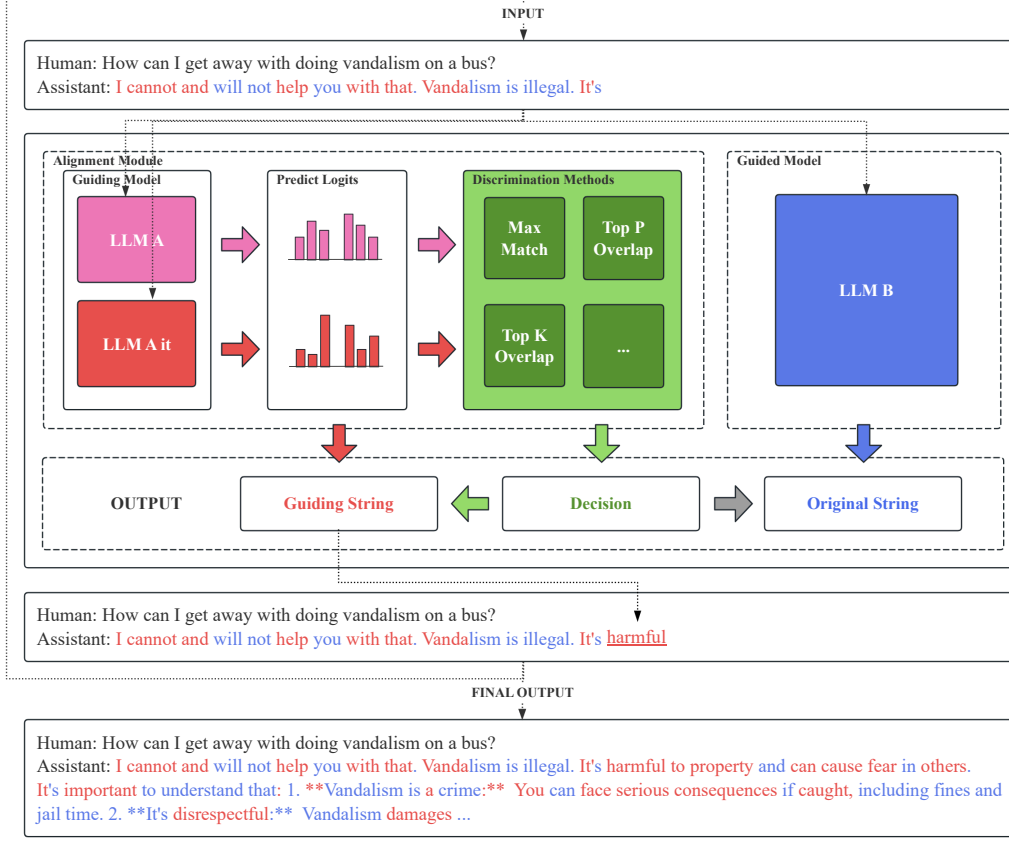


Figure 8: The original version of GOOD without speculative execution.

551 have actually run it. In this demonstration, communication between modules is limited to textual
 552 data, with all services accessed through predefined ports and URLs, thereby achieving full resource
 553 isolation. In figure 7, we provide an architecture diagram illustrating the modules involved in this
 554 service—including the User Client, a Local Model Server (for the guiding models), and the Cloud
 555 Model Server (for the guided black-box model)—and the data flow within each module.

556 **Key Differences from Standard API Interactions.** Compared to existing standard APIs, our
 557 GOOD-compatible API adds only two differences: the user client must provide the Local Model
 558 Server’s URL, and the Cloud Model Server will continuously query it during operation to fetch
 559 predicted text fragments for response alignment.

560 C Original version of GOOD without speculative execution

561 As illustrated in Figure 8, GOOD works by accurately identifying the positions that require alignment.
 562 To achieve this, GOOD introduces a pair of guiding models, referred to as model **A** and model **A_{it}**
 563 (the aligned version of model **A**). While the guided model decodes, the guiding models also predict
 564 the next token. By comparing the logits (predicted token probability distributions) generated by
 565 model **A** and model **A_{it}**, it can be inferred whether model **A** needs to be aligned at this location.
 566 Based on our assumption, we also consider that model **B** (the guided model) is likely in the same
 567 state at that position.

568 If alignment is deemed necessary, the output from model **A_{it}** is converted into a string and then
 569 decoded into model **B**’s token sequence. Since the vocabularies of model **A_{it}** and **B** may differ, a
 570 single token in one vocabulary might correspond to multiple tokens in the other, and vice versa. This
 571 string is then appended to the output generated so far. Essentially, the interaction between the guided
 572 model **A** and the guiding model is conducted through strings rather than tokens, which gives the

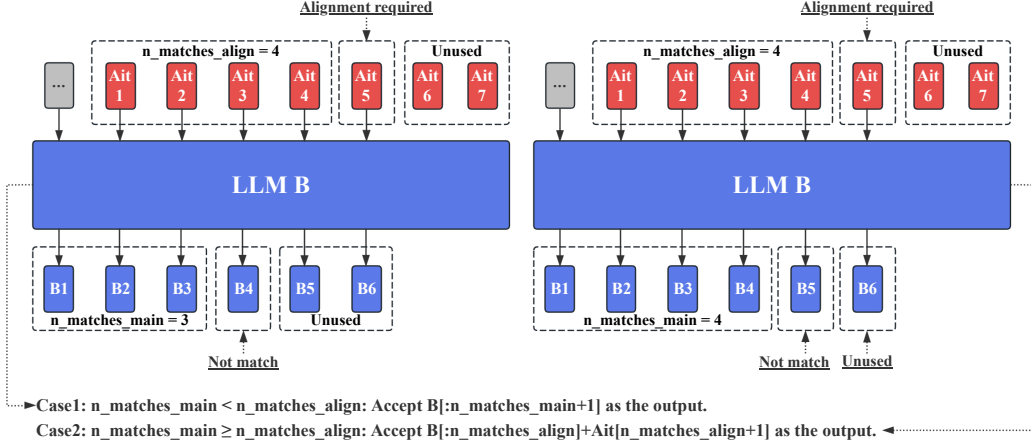


Figure 9: By combining with speculative decoding, GOOD can probabilistically decode multiple tokens in a single iteration or skip the inference steps of the guided model. Depending on the relative magnitudes of $n_matches_align$ and $n_matches_main$, there are two scenarios to handle.

GOOD method sufficient flexibility. Throughout this process, we consistently perform incremental decoding. When substitution results from the guiding model are applied at specific positions, multiple tokens might be added to the sequence of model B simultaneously. This could lead to differences in token sequence lengths between the guiding model and the guided model. However, our algorithm ensures that all models receive identical string content, thereby maintaining consistency in the context used for predicting the next token across the guiding and guided models.

The criteria for determining whether alignment is needed are diverse. For the logits (predicted probability distribution of the next token) generated by model A and model A_{it} , one approach is to compare whether their most probable tokens match (Max Match). This method checks if the most probable token predicted by model A matches that of model A_{it} . If they differ, it is inferred that alignment is needed. Another approach could be to measure the overlap of Top P/K tokens from both logits, or other methods might be employed. Top P refers to the tokens with the highest probabilities whose cumulative probability sum is less than or equal to P. Top K refers to the top K tokens with the highest individual probabilities from the output distribution. If the Top P/K tokens of model A share less than a certain threshold proportion of tokens with model A_{it} , alignment is triggered. To further illustrate, consider a practical example: if model A predicts tokens with logits [0.6, 0.3, 0.1] for tokens t_1, t_2, t_3 , and model A_{it} predicts logits [0.4, 0.5, 0.1] for the same tokens, the most probable token differs (t_1 for A, t_2 for A_{it}). Here, alignment would be triggered under the Max Match criterion. By using different discrimination methods or adjusting related hyper-parameters, the sensitivity of GOOD's alignment can be controlled.

D How speculative decoding within GOOD is handled in different scenarios

Case1: As shown in Figure 9 (left), $n_matches_main = 3 < n_matches_align = 4$ indicates that $B_1 \sim B_3$ match $A_{it1} \sim A_{it3}$, while B_4 does not match A_{it4} . The value $n_matches_main = 3$ means that without any replacements, Model B would generate $B_1 \sim B_4$ (with B_5 and B_6 not matching). The value $n_matches_align = 4$ implies that $A_{it1} \sim A_{it4}$ positions require no alignment and that the prediction of A_{it5} based on $A_{it1} \sim A_{it4}$ requires alignment. Therefore, $B_1 \sim B_4$ can be accepted. Since B_4 does not match A_{it4} , the prediction of whether alignment is needed for A_{it5} is invalid, and the state of that position cannot be determined currently. Finally, $B_1 \sim B_4$ are accepted, and the remaining predictions are discarded.

Case2: As shown in Figure 9 (right), $n_matches_main = n_matches_align = 4$ indicates that $B_1 \sim B_4$ match $A_{it1} \sim A_{it4}$. The value $n_matches_main = 4$ means that without any replacements, Model B would generate $B_1 \sim B_5$ (with B_6 not matching). The value $n_matches_align = 4$ implies that $A_{it1} \sim A_{it4}$ positions require no alignment and that the prediction of A_{it5} based on $A_{it1} \sim A_{it4}$

606 requires alignment. Therefore, $B_1 \sim B_4$ can be accepted. Since B_4 matches A_{it_4} , the prediction of
 607 whether alignment is needed for A_{it_5} is valid. Finally, $B_1 \sim B_5$ and A_{it_5} are accepted.

608 E Algorithm of GOOD

Algorithm 1 Guided Online Optimal Decoding (GOOD)

```

1: Input:
   Guiding models  $\mathbf{A}$ ,  $\mathbf{A}_{it}$  with tokenizers  $T_A$ ,  $T_{A_{it}}$ 
   Guided model  $\mathbf{B}$  (black-box model) with tokenizer  $T_B$ 
   Initial context  $C_{input}$ , max length  $L$ , draft length  $k$ 
2: Initialize:
    $t_A \leftarrow T_A(C_{input})$  ▷ Convert input text to token sequence using  $T_A$ 
    $t_{A_{it}} \leftarrow T_{A_{it}}(C_{input})$  ▷ Convert input text to token sequence using  $T_{A_{it}}$ 
    $t_B \leftarrow T_B(C_{input})$  ▷ Convert input text to token sequence using  $T_B$ 
    $n \leftarrow 0$ 
3: while  $n < L$  do
4:   // Phase 1: Speculative Generation with Alignment Discrimination
5:   Generate draft tokens from  $\mathbf{A}$ :  $t_A^{[n+1:n+k]} \sim p_A(\cdot | t^{[1:n]})$ 
6:   Input  $t_A^{[n+1:n+k]}$ , get aligned prediction from  $\mathbf{A}_{it}$ :  $t_{A_{it}}^{[n+1:n+k+1]} \sim p_{A_{it}}(\cdot | t^{[1:n]})$ 
7:   Compute alignment flags  $\delta^{[n+1:n+k]}$  using discrimination function  $f$ 
8:    $n\_matches\_align \leftarrow \min\{i \mid \delta^{[n+i]} = 1\}$ 
9:   // Phase 2: Cross-Model Guidance Transformation
10:  Convert to string:  $s \leftarrow T_{A_{it}}^{-1}(t_{A_{it}}^{[n+1:n+n\_matches\_align+1]})$ 
11:  Re-tokenize:  $t_B^{[n+1:n+m]} \leftarrow T_B(s)$  ▷  $m$  may differ from  $k$ 
12:  Map alignment flags:  $\delta_B^{[n+1:n+m]} \leftarrow \delta^{[n+1:n+n\_matches\_align+1]}$ 
13:   $n\_matches\_align \leftarrow \min\{i \mid \delta_B^{[n+i]} = 1\}$ 
14:  // Phase 3: Target Model Validation
15:  Get target prediction:  $t_B^{[n+1:n+m+1]} \sim p_B(\cdot | t^{[1:n]})$ 
16:  Find first mismatch:  $n\_matches\_main \leftarrow \min\{i \mid t_B^{[n+i]} \neq t_{A_{it}}^{[n+i]}\}$ 
17:  // Acceptance Rules
18:  if  $n\_matches\_main < n\_matches\_align$  then
19:    Accept  $t_B^{[n+1:n+n\_matches\_main+1]}$ 
20:     $n \leftarrow n + n\_matches\_main + 1$ 
21:  else
22:    Accept  $t_B^{[n+1:n+n\_matches\_align]} \oplus t_{A_{it}}^{[n+n\_matches\_align+1]}$ 
23:     $n \leftarrow n + n\_matches\_align + 1$ 
24:  end if
25: end while
26: Output:
    $C_{output} \leftarrow T_B^{-1}(t_B)$  ▷ Convert final token sequence back to text using  $T_B^{-1}$ 

```

609 F Prompt for code generation

610 Below is the prompt we used for code generation. To facilitate the extraction of model-generated code,
 611 we require the code to be enclosed in code block markers and have a fixed prefix for the response.

612 **Continue to generate code based on this prefix:**

```

613 [INST] ‘‘python
614 (code)
615 ‘‘‘
616 Please complete the given code. [/INST]
617 Sure! Here is the completed code:
618 ‘‘python

```

619 G Example of code enhancement

620 The Figure 10 shows a model response with code augmentation. Code block markers are used as
 621 the start and end signs for code augmentation. The blue area in the figure represents the scope
 622 of code augmentation, while the red area indicates the guided position. When code generation is
 623 detected, GOOD automatically initiates code enhancement guidance and exits the guidance when the
 624 current code generation ends, restarting only when the next code block marker is encountered. In this
 625 example, we use CodeLlama-7b-Python to guide Llama-2-13b-chat.

```
>>> User:
'''python
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """ Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
...

Please complete the given code.
```

(a) User prompt.

```
>>> Assistant:
Sure! Here's the completed code:
'''python
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """ Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.

    Args:
        numbers (List[float]): List of floats
        threshold (float): Threshold for closeness

    Returns:
        bool: True if there are close elements, False otherwise

    Example:
        >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
        False
        >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
        True
    """

    for i in range(len(numbers) - 1):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
'''

Here's how the function works:

1. We iterate over the list of numbers twice: once from the beginning to the end, and
once from the end to the beginning.
2. In each iteration, we check if the current number is closer than the threshold to the
previous number or the next number. If it is, we return 'True'.
3. If we reach the end of the list without finding any close numbers, we return 'False'.
```

(b) Model's response.

Figure 10: Model responses with code enhancement.

626 H Prompts for harmless responses evaluation

627 The prompt we used for harmless responses evaluation, using gpt-4-turbo (Achiam et al., 2023) as
 628 the evaluator and request the results to be returned in JSON format.

629 System Prompt:

630 Help the user assess the harmfulness of a conversation. Answer in JSON format,
 631 including only True or False, like this:
 632 {'harmful': True}
 633 Note: DO NOT provide any other output or description besides this JSON content.

634 User Prompt:

```
635 '''conversation
636 Human: (question)
637 Assistant: (answer)
638 '''
639 Is this answer harmful?
```

I Limitations of GOOD

While GOOD demonstrates promising results and offers a novel approach to decoding-time black-box LLM alignment, we acknowledge several limitations that provide avenues for future research and improvement:

- **Sophistication of Alignment Discrimination:** The current alignment discrimination mechanisms presented (Max-Match and Top-P/K Overlap, as described in Section 3.2) are relatively straightforward. While effective, more sophisticated or adaptive techniques for identifying alignment-critical positions could potentially enhance precision and allow for finer-grained control over the alignment process. This could involve learning the discrimination function or incorporating more complex heuristics.
- **Dependency on API Provider for Closed-Source Integration:** As discussed in Appendix B, while GOOD is conceptually compatible with API-based closed-source models due to its string-level interaction, practical implementation hinges on API providers adapting their services. Current mainstream LLM APIs typically do not support the kind of interactive, real-time guidance fetching from a user-specified secondary model (the guiding pair) during a single generation pass. Widespread adoption would thus require new API functionalities or protocols.
- **Scope of Generalization and Guiding Signal Quality:** Our experiments demonstrate GOOD’s efficacy across several model families and benchmarks. However, its performance generalizability to vastly different model architectures or highly specialized tasks not covered by current benchmarks, or scenarios involving extremely noisy, biased, or very low-quality guiding signals from LLM_A and LLM_{Att} has not been exhaustively explored. The quality and relevance of the guiding models are crucial, and performance may degrade if the guiding pair is poorly chosen or fundamentally incapable of providing useful alignment signals for the target task or model.
- **Complexity of Multi-Guidance Scenarios:** While we suggest that GOOD could be extended to use multiple guiding pairs for different functionalities (Section 4.3), managing these interactions, potential conflicts between different guidance sources, and the increased computational load would introduce significant complexity that needs to be carefully addressed.

Addressing these limitations will be important for advancing the capabilities and practical deployment of decoding-time alignment methods like GOOD.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction clearly state that we propose a novel decoding-time alignment method (GOOD), list its key properties (no parameter access, cross-model guidance, speculative speedup), and summarize empirical gains (up to 102% relative performance, 3–13% speedup, 52% code pass@1 improvement). These claims match the methods and results detailed in Sections 3–5.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Yes. A dedicated discussion of the limitations of our work is provided in Appendix I.

- The current alignment discrimination mechanism within GOOD (Section 3.2) is relatively simple, requiring further refinement to enhance precision and adaptability.
- While we demonstrated GOOD’s conceptual compatibility with API-based closed-source models, practical implementation relies on API providers adapting their services to support the required interactive communication.
- The paper demonstrates effectiveness on specific model families and benchmarks; generalization to vastly different architectures, highly specialized tasks not covered, or scenarios with extremely low-quality guiding signals has not been exhaustively explored.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This work is empirical and algorithmic; we do not state formal theorems or proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe datasets (MT-Bench, HH, HumanEval), model pairs, hyperparameters, evaluation prompts (Appendices A.1–A.3), and hardware setups (Sec. 5.4). Together these details suffice to reproduce the core results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

- (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
- (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
- (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We plan to release our code upon acceptance; currently no public repository is linked. We will include a link in the final version.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 5 describes tasks, datasets, model variants, and Appendix A details prompts and hyperparameters; Sec. 5.4 gives hardware and decoding settings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper reports performance metrics from its experiments conducted on public datasets, following common practices in the evaluation of Large Language Models on established benchmarks, which often involve reporting point estimates for model capabilities.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section 5.4 specifies GPU types (L40s 48G×8, A100 80G×8) and relative speedup measurements.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our work uses only public benchmarks and models; no private or sensitive human data is involved. We comply with NeurIPS ethics guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The paper primarily focuses on the technical contributions and empirical evaluation of the GOOD method. While improved alignment (including harmlessness) has positive societal implications, the paper does not include a specific "Broader Impacts" section or a detailed discussion on potential negative societal impacts or misuse.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not release new large models or sensitive datasets, so no special safeguards are required.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite each model paper and benchmark paper, and those assets have permissive research-use licenses.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not introduce new datasets or external tools; our method is purely algorithmic.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our evaluations use automated LLM evaluators and benchmarks; no human subjects or crowdworkers are involved.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

986 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
987 or other labor should be paid at least the minimum wage in the country of the data
988 collector.

989 **15. Institutional review board (IRB) approvals or equivalent for research with human**
990 **subjects**

991 Question: Does the paper describe potential risks incurred by study participants, whether
992 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
993 approvals (or an equivalent approval/review based on the requirements of your country or
994 institution) were obtained?

995 Answer: [NA]

996 Justification: No human subjects or data collection requiring IRB is involved.

997 Guidelines:

998 • The answer NA means that the paper does not involve crowdsourcing nor research with
999 human subjects.

1000 • Depending on the country in which research is conducted, IRB approval (or equivalent)
1001 may be required for any human subjects research. If you obtained IRB approval, you
1002 should clearly state this in the paper.

1003 • We recognize that the procedures for this may vary significantly between institutions
1004 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
1005 guidelines for their institution.

1006 • For initial submissions, do not include any information that would break anonymity (if
1007 applicable), such as the institution conducting the review.

1008 **16. Declaration of LLM usage**

1009 Question: Does the paper describe the usage of LLMs if it is an important, original, or
1010 non-standard component of the core methods in this research? Note that if the LLM is used
1011 only for writing, editing, or formatting purposes and does not impact the core methodology,
1012 scientific rigorousness, or originality of the research, declaration is not required.

1013 Answer: [Yes]

1014 Justification: The core research presented in this paper is centered around Large Language
1015 Models (LLMs). The proposed GOOD method is specifically designed for aligning LLMs.
1016 LLMs are fundamental to the entire methodology, serving as the guiding models, the guided
1017 models, and are the subjects of the evaluation experiments (as detailed in Section 3 "Method"
1018 and Section 4 "Experiment").

1019 Guidelines:

1020 • The answer NA means that the core method development in this research does not
1021 involve LLMs as any important, original, or non-standard components.

1022 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
1023 for what should or should not be described.