Efficient Model-agnostic Alignment via Bayesian Persuasion

Fengshuo Bai 1,2,3 Mingzhi Wang 2,†* Zhaowei Zhang 2,3,† Boyuan Chen 2,† Yinda Xu 1 Ying Wen 1,‡ Yaodong Yang 2,‡

¹Shanghai Jiao Tong University
²Institute for Artificial Intelligence, Peking University
³National Key Laboratory of General Artificial Intelligence, BIGAI

Abstract

With recent advancements in large language models (LLMs), alignment has emerged as an effective technique for keeping LLMs consensus with human intent. Current methods primarily involve direct training through Supervised Fine-tuning (SFT) or Reinforcement Learning from Human Feedback (RLHF), both of which require substantial computational resources and extensive ground truth data. This paper explores an efficient method for aligning black-box large models using smaller models, introducing a model-agnostic and lightweight Bayesian Persuasion Alignment framework. We formalize this problem as an optimization of the signaling strategy from the small model's perspective. In the persuasion process, the small model (Advisor) observes the information item (i.e., state) and persuades large models (Receiver) to elicit improved responses. The Receiver then generates a response based on the input, the signal from the Advisor, and its updated belief about the information item. Through training using our framework, we demonstrate that the Advisor can significantly enhance the performance of various Receivers across a range of tasks. We theoretically analyze our persuasion framework and provide an upper bound on the Advisor's regret, confirming its effectiveness in learning the optimal signaling strategy. Our Empirical results demonstrates that GPT-2 can significantly improve the performance of various models, achieving an average enhancement of 16.1% in mathematical reasoning ability and 13.7% in code generation. We hope our work can provide an initial step toward rethinking the alignment framework from the Bayesian Persuasion perspective.

1 Introduction

Recent years have witnessed increased attention and effort in aligning large language models (LLMs) with human intentions and values [35, 26]. This alignment is facilitated by providing reliable supervision through demonstrations [8, 45], reward signals [37], preferences [16, 40] or critiques [42, 5], and by employing methods such as supervised learning (*e.g.*, Supervised Fine-tuning, SFT) or reinforcement learning (*e.g.*, Reinforcement Learning from Human Feedback, RLHF) [37].

However, these methods, including RLHF, require multiple models and direct training of large models, which significantly increases computational demands [40]. Moreover, Fine-tuning cannot be applied to closed-source models, complicating output control for alignment with human intents. Additionally, current alignment methods, like SFT and RLHF, face limitations when human evaluators lack expertise in complex tasks [7, 43]. These challenges highlight the need for efficient, scalable alignment strategies for both open-source and closed-source models. Therefore, our work aims to address the above challenges by answering the following question:

^{*†} Equal Contribution. [‡]Correspondence to Yaodong Yang<*yaodong.yang@pku.edu.cn>* and Ying Wen<*ying.wen@sjtu.edu.cn>*.

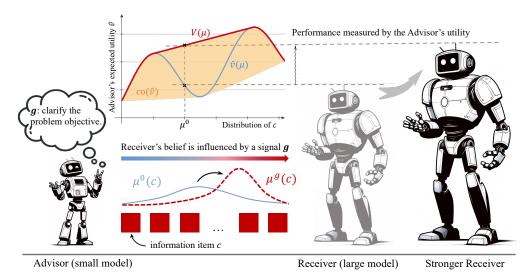


Figure 1: An illustration of our persuasion framework. The Receiver observes a signal g sent by the Advisor and updates its belief from the prior distribution μ^0 to a posterior distribution μ^g . The axes depict the Advisor's expected utility $\hat{v}(\mu)$ across various information distributions μ . In this context, $co(\hat{v})$ denotes the convex hull of \hat{v} , while V represents the concave closure of \hat{v} . Here, $V(\cdot)$ is the largest expected utility Advisor can achieve with any signal. From the Advisor's perspective, the Receiver's performance is enhanced following persuasion.

Can we use a smaller model to influence the behaviors of larger models, thereby enabling alignment and enhancing performance with little human supervision or feedback?

Inspired by Bayesian persuasion [30], we model the alignment problem as an information design process, involving a protocol between a small model and a large model. Instead of training multiple models as in techniques like RLHF, we only employ a small model with minimal supervision to learn a signaling strategy that influences the behaviors of fixed large models. In this setup, large models are treated as a black box. This modeling approach significantly reduces the demand for computational resources by delegating alignment tasks to smaller models, leading to a more lightweight and efficient framework. This makes it inherently suitable for a broader range of alignment scenarios, accommodating tasks of varying difficulty and larger models.

In this work, we introduce a novel framework termed Bayesian Persuasion Alignment, as illustrated in Figure 1. We frame alignment as a Bayesian Persuasion (BP) problem, wherein a smaller model serves as the Advisor and a larger model acts as the Receiver. In this setup, the Advisor generates a signal that is sent to the Receiver. Upon receiving this signal, the Receiver updates its beliefs and produces a response. Our core insight is that a smaller model trained on supervision for optimal signaling strategy can effectively *persuade* larger models, thereby improving the quality of their responses, *i.e.*, their outputs. This mechanism provides several advantages: (1) From the perspective of information design, the well-defined signaling strategy of the Advisor ensures an increase in the Advisor's utility without decreasing the Receiver's utility [30]. (2) The Advisor manipulates the Receiver's belief to enhance performance, significantly reducing the need for training resources while ensuring alignment performance, making it an effective and parameter-efficient alignment strategy. (3) Moreover, the learned signaling strategy can be applied to different Receivers, guaranteeing a model-agnostic nature and making it easier to generalize to harder tasks.

To the best of our knowledge, BP Alignment is the first integration of Bayesian persuasion with the alignment framework. Our main contributions can be summarized in three folds: **First**, we introduce a parameter-efficient and model-agnostic alignment framework that trains a smaller model to enhance the performance of various larger models. **Second**, we demonstrate that our persuasion framework significantly improves the performance of various large models on mathematical problem-solving and code-generation tasks. Specifically, the Advisor (Phi-2) enables significant enhancements, with an average improvement of 22.5% on GSM8K [17], 39.0% on MATH [23], and a 24.7% increase on

HumanEval [12]. Lastly, we theoretically analyze our framework and provide an upper bound on the Advisor's regret, indicating its effectiveness in learning the optimal signaling strategy.

2 Related Work

In this section, we will review existing works on Scalable Oversight, AI Persuasion, and Eliciting Latent Knowledge (ELK).

Scalable Oversight. As models begin to achieve broadly human-level performance and take on more complex tasks that are difficult for humans to understand, providing continual, reliable feedback and ensuring that the models' behaviors align with human intents becomes challenging. This naturally raises the crucial issue of scalable oversight: how can we provide supervisory signals to more powerful AI systems and ensure they are aligned with human intents? [36, 2, 26] Unlike current methods that focus on enhancing the capabilities of weak supervisors [14, 7, 31], our framework addresses this challenge by transforming weak supervisors into persuaders and identifying the optimal signal-sending strategy to effectively influence the behaviors of stronger models.

Our work also differs from weak-to-strong generalization [10] and similar alignment methods [27, 32]. These methods face a trade-off: the strong model may either mimic the weak model, reducing performance or use its reasoning abilities to improve [10]. Additionally, they often rely heavily on ground truth labels. In contrast, our approach uses small models as Advisors to elicit the capabilities of stronger models without adding noisy labels. Guided by the information design principle, our method scales to various stronger models and challenging tasks, minimizing the need for ground truth. We evaluate our method on various mathematical problem-solving and code-generation tasks using only GPT-2 as the Advisor. It achieves significant advancements, with an average improvement of 9.5% on GSM8K, 22.6% on MATH, and 13.7% on HumanEval across a range of strong models.

AI Persuasion. Persuasion is a dynamic game process in which one player (sender) influences the beliefs or actions of another player (receiver) by providing informative signals, thereby affecting the outcomes for both players. AI persuasion can be categorized into two types based on the target: (1) AI persuading humans to change their original viewpoints (*i.e.*, Captology) [21, 47, 33, 19], and (2) employing persuasive signals to change the behavior of AI systems. While most existing research has concentrated on the former, studies on the latter remain relatively nascent. In this paper, we primarily explore the latter category.

Zeng et al. [48] conducted a comprehensive review of decades of social science research and proposed a taxonomy to automatically generate persuasive adversarial prompts that induce unsafe behaviors in LLMs. However, their method lacks a formal definition or analysis of persuasive behavior and its impact. Bayesian persuasion [30, 29] is a symmetric information framework that utilizes to influence beliefs by strategically sharing information aligned with motivations, aiding decision-making tasks [22]. While its application to language models remains unexplored, [49] suggested it could align AI during deployment by tailoring information based on different scenarios. To the best of our knowledge, we are the first to use dialogue to apply Bayesian persuasion to LLMs and enhance their capabilities.

Eliciting Latent Knowledge. Christiano et al. [15], Hobbhahn [24] introduced a theoretical framework termed Eliciting Latent Knowledge (ELK), designed to extract latent knowledge from models to assess whether AI systems align with human intents. This framework includes aspects such as honesty analysis [20], knowledge elicitation [9, 38], and general task knowledge acquisition [10]. Our framework utilizes advisor persuasion to elicit strong models' latent knowledge for solving difficult tasks and has the potential for honesty analysis.

Typical ELK methods struggle to train models to report true beliefs rather than just aligning with human preferences. This issue arises because both strategies yield identical training losses, as they produce the same answers to training inputs [24]. As a result, models tend to align with human expectations instead of reporting their own beliefs. Our method addresses this by decoupling the training objective into a reporting objective and a persuasion objective, focusing on optimal signaling rather than human-evaluated ground truth.

3 Bayesian Persuasion Alignment

In this section, we formally introduce the proposed persuasion framework. We begin by establishing the notations and outlining the persuasion protocol, followed by defining the overall objective and conducting a theoretical analysis of regret.

3.1 Protocol and Notations

We introduce the persuasion protocol wherein an Advisor (small model) persuades a Receiver (large model) to improve its response to a given input. For each input x, a finite set \mathcal{C}_x contains all associated information items (i.e., state) with input x. The Receiver's utility function u(x,c,y) is continuous and dependent on its response $y \in \mathcal{Y}$ to the input $x \in \mathcal{X}$ and the associated information item $c \in \mathcal{C}_x$. Similarly, the Advisor has a continuous utility function v(x,c,y), which is contingent on the Receiver's response, input, and associated information item. Importantly, in our settings, the Advisor lacks knowledge of the Receiver's utility function. For each input x, the Advisor and Receiver start with a shared prior $\mu_x^0 \in \operatorname{int}(\Delta(\mathcal{C}_x))^2$. The signaling strategy π is defined by a finite realization space \mathcal{G} and a family of distributions $\pi_x(\cdot|c), c \in \mathcal{C}_x$ over \mathcal{G} . This strategy is implemented through a neural network with parameters θ . The Advisor sends a signal, and the Receiver observes the chosen signal realization $g \in \mathcal{G}$ (with $|\mathcal{G}| < \infty$).

The game timing in persuasion is as follows: The Advisor commits to a signaling strategy π_{θ} and announces it to the Receiver. For a given input x, the Advisor observes an information item sampled from μ_x^0 and sends a signal g to the Receiver. Upon receiving this signal, the Receiver updates its belief about information item in \mathcal{C}_x , forming posterior distribution μ_x^g via Bayes's rule. The Receiver then chooses a response from the response set, which is defined by

$$y^*(\mu_x^g) = \arg\max_{y \in \mathcal{Y}} \underset{c \sim \mu_x^g}{\mathbb{E}} \left[u(x, c, y) \right]. \tag{1}$$

The solution of the game is the problem of optimal signaling strategy design from the Advisor's point of view. Taking the Receiver's response as given, the Advisor selects a signaling strategy π_{θ} that maximizes its expected utility. Since the responses are generated only by the Receiver, the Advisor cannot directly influence the information set. Instead, the Advisor can leverage its informational advantage concerning the information item to influence the Receiver indirectly by way of signaling, thereby persuading the Receiver to generate improved responses.

3.2 Signaling Strategy and Belief Update

A signaling strategy of the Advisor generates a distribution over \mathcal{G} , which is the signal realization space. Formally, the signaling strategy π_{θ} comprises a function $\pi_x : \mathcal{C} \to \Delta(\mathcal{G})$ for each input $x \in \mathcal{X}$. Upon observing an information item c, the Advisor sends a signal sampled from $\pi_x(c)$, where the input is x, and $\pi_x(g,c)$ denotes the probability of $g \in \mathcal{G}$ within this distribution.

The signal space \mathcal{G} is broadly construed, including some uninformative signaling strategies. For instance, the Advisor may send the same signal regardless of the information item c, such that $\pi_x(c) = \pi_x(c')$ for all $c, c' \in \mathcal{C}_x$. Without loss of generality, we assume that signals in \mathcal{G} are perceived as distinct by the Receiver.

Upon receiving a signal g, the Receiver updates its posterior belief regarding the information items. The conditional probability of the information item being c is defined as:

$$\Pr(c|g, \pi_x) = \frac{\mu_x(c)\pi_x(g|c)}{\sum_{c' \in \mathcal{C}_x} \mu_x(c')\pi_x(g|c')}.$$
 (2)

The derivation of the Receiver's posterior belief also depends on its knowledge of the signaling strategy π_{θ} . In accordance with the Bayesian persuasion framework, the Advisor commits to a signaling strategy π_{θ} at the beginning of the process and announces it to the Receiver.

3.3 Signaling Strategy Optimization

From the Advisor's perspective, the objective is to identify a signaling strategy π_{θ} that maximizes the Advisor's expected utility, thereby inducing superior responses from the Receiver. Accordingly, the

 $^{^2}$ int(X) denotes the interior of the set X and $\Delta(X)$ represents the set of all probability distributions on X

signaling strategy is optimized by minimizing the following loss function:

$$\mathcal{L}(\theta) = - \underset{x \in \mathcal{X}}{\mathbb{E}} \left[\sum_{c \sim C_x} \Pr(c|g, \pi_x) \ v(x, c, y) \right], \tag{3}$$

where y is the Receiver's response to input x as determined by equation (1).

3.4 Regret Analysis

Although the Bayesian Persuasion Alignment framework we propose is practically appealing, a pivotal theoretical question arises: *How can we ensure that this framework robustly learns the optimal signaling strategy over time?* Specifically, it is crucial to demonstrate that the Advisor can gradually find the most persuasive signaling strategy through its interactions with the Receiver. This convergence guarantee is essential for our framework. To address this question, we draw inspiration from the online Bayesian persuasion setting [11, 6] and analyze the performance of our algorithm from an online learning perspective. We introduce the concept of regret, which quantifies the utility difference between the algorithm's performance and the optimal strategy over a certain period. Demonstrating that the algorithm's regret grows sublinearly with time would imply that it can progressively converge to the optimal strategy.

Without loss of generality, we focus on signaling schemes that are both *direct* and *persuasive* [3], according to the revelation principle. In a *direct* signaling scheme, the signals directly correspond to response recommendations for the Receiver. Moreover, a signaling scheme is considered *persuasive* if it incentives the Receiver to follow the response recommendations provided by the Advisor. Let $\mathcal P$ denote the set of *direct* and *persuasive* signaling schemes, where each element $\phi \in \mathcal P$ is a mapping $\phi: \mathcal C \to \Delta(\mathcal Y)$. To simplify the notation, we omit x in subsequent analysis. With the definition of the set of persuasive signaling schemes $\mathcal P$, the Advisor's expected utility under a signaling scheme $\phi \in \mathcal P$ can be expressed as follows:

$$v(\phi) := \sum_{c \in \mathcal{C}} \sum_{y \in \mathcal{Y}} \mu_c \phi_c(y) v(c, y), \tag{4}$$

where μ_c represents the prior probability of information item c, $\phi_c(y)$ denotes the probability that the signaling scheme ϕ recommends response y under information item c, and v(c,y) is the Advisor's utility when the Receiver takes response y under information item c.

Next, we introduce a linear mapping f that maps each signaling scheme $\phi \in \mathcal{P}$ to a point in the $\mathbb{R}^{|\mathcal{Y}|}$ space. Specifically, for each $\phi \in \mathcal{P}$, we define

$$f(\phi) := \left[-v(\phi, y) \right]_{y \in \mathcal{V}},\tag{5}$$

where $v(\phi,y)=\sum_{c\in\mathcal{C}}\mu_c\phi_c(y)v(c,y)$ represents the Advisor's expected utility when the Receiver takes response y under signaling scheme ϕ .

Intuitively, the mapping f represents each signaling scheme as a $|\mathcal{Y}|$ -dimensional vector, where each component $-v(\phi,y)$ represents the negative of the Advisor's expected utility for response y. This representation embeds the signaling schemes into a Euclidean space that directly corresponds to the Receiver's response space. Furthermore, we examine the convex hull of the graph of f, denoted as $\operatorname{co} f(\mathcal{P})$. Each point within $\operatorname{co} f(\mathcal{P})$ corresponds to a convex combination of signaling schemes.

Formally, the Advisor's regret at round T is defined as:

$$R_T := \max_{\phi \in \mathcal{P}} \sum_{t=1}^{T} v(\phi) - \sum_{t=1}^{T} v(\phi_t).$$
 (6)

To analyze the regret bound of our persuasion framework, we present the theoretical version of our algorithm in Algorithm 1.

Algorithm 1 Theoretical Persuasion Algorithm

```
Require: Set of information items \mathcal{C}, set of responses \mathcal{Y}, prior distribution \mu^0, Advisor's utility
      function v, regret minimizer \mathcal{R} for the set co f(\mathcal{P})
 1: for t = 1, ..., T do
            \operatorname{co} f(\mathcal{P}) \ni z_t \leftarrow \mathcal{R}.\mathsf{RECOMMEND}()
           \{(\phi_t^{(i)}, \lambda_t^{(i)})\}_{i \in [m+1]} \leftarrow \mathsf{DECOMPOSE}(z_t, f(\mathcal{P})) Sample i_t \in [m+1] with probabilities \lambda_t^{(1)}, \dots, \lambda_t^{(m+1)}
 3:
                                                                                                                4:
           Let \phi_t \leftarrow \phi_t^{(i_t)}
 5:
            Observe information item c_t \sim \mu^0
 6:
 7:
            Select and play action y_t \sim \phi_t(\cdot|c_t)
            \mathcal{R}.\mathsf{OBSERVE}(v(c_t, y_t))
 8:
 9: end for
```

The key idea behind this algorithm is to maintain a regret minimizer \mathcal{R} over the possible signaling strategies, represented by the convex hull of the set of posterior distributions $\operatorname{co} f(\mathcal{P})$. At each round t, the algorithm obtains a recommended strategy z_t from \mathcal{R} and decomposes it into a convex combination of extreme points $\{(\phi_t^{(i)}, \lambda_t^{(i)})\}_{i \in [m+1]}$ using Caratheodory's Theorem [18]. The algorithm then samples an index i_t according to the weights $\lambda_t^{(i)}$ and plays the corresponding signaling strategy $\phi_t = \phi_t^{(i_t)}$. Upon observing the realized information item c_t and the Advisor's utility $v(c_t, y_t)$ for the chosen reponse y_t , the algorithm updates the regret minimizer with this feedback.

Under this theoretical algorithm, we can derive the following regret bound.

Theorem 1. Algorithm 1 guarantees regret $R_T = O(m^{3/2}\sqrt{T \log T})$, where $m = |\mathcal{Y}|$ is the number of receiver's reponses.

The regret bound presented in Theorem 1 demonstrates that our algorithm achieves sublinear regret over the time horizon T, with a dependence on the size of Receiver's response space m. The output space of LLMs is theoretically infinite, as they can generate text of arbitrary length. However, each response's length is practically limited. Additionally, responses with same semantics are considered equivalent given a specific input. Therefore, the Receiver's response space can be regarded as finite, aligning well with the assumptions in our theoretical analysis. Although there are differences between the theoretical algorithm and its practical implementation, the core principle of learning the optimal signaling strategy through interaction remains consistent. This consistency provides a theoretical guarantee for the algorithm's performance and demonstrates its effectiveness in learning the optimal signaling strategy over time.

4 Experiments

In this section, we evaluate the effectiveness of our persuasion framework on mathematical problems and code generation. Our evaluation aims to address the following key questions:

- (1) Can our framework enhance the Receiver's performance in various tasks? (Section 4.2.1)
- (2) Can our framework find a non-trivial signaling strategy? (Section 4.2.2)
- (3) How about the efficiency of the proposed framework? (Section 4.2.4)

Furthermore, we investigate the generalization of the signaling strategy across different Receivers (Section 4.2.1), across varying difficulties (Section 4.2.3), and for various tasks (Appendix B.1). Details on experiments are provided in Appendix A.

4.1 Settings

Implementation Details. We train the Advisor for math-solving tasks using the training datasets from GSM8K and MATH, and for the code generation task using the training dataset from MBPP. We construct an information set for each input using Llama3-8B-Instruct ³ for all datasets. Specifically, each input includes seven information items, each emphasizing different key aspects essential for problem-solving. Further details on information set construction are provided in Appendix A.1. In

³https://github.com/meta-llama/llama3

Table 1: **Performance of various Receivers under persuasion.** We report the accuracy on GSM8K and MATH, and the pass@1 score on HumanEval across four information structures. "Posterior Information" refers to sampling the information item from the posterior distribution, influenced by the Advisor. The Advisor for math tasks differs from that for code generation tasks. Arrows indicate performance improvements relative to the prior distribution.

Task	Receiver	No Information	All Information	Prior Information	Posterior Information	
					Advisor (GPT-2)	Advisor (Phi-2)
GSM8K (8-shot)	Phi-2	56.0	41.0	56.8	59.1	62.1
	Mistral-7B	34.3	48.0	45.7	50.4	53.8
	Llama2-7B	15.1	36.6	27.2	34.5	45.6
	Llama2-7B-Chat	21.8	31.8	37.3	40.0	50.0
	Llama2-13B	25.2	38.9	36.2	38.9	45.9
	Llama2-13B-Chat	33.9	37.3	36.1	37.9	39.2
	Llama3-8B	47.6	54.0	53.7	56.0	62.6
	Llama3-8B-Instruct	73.5	72.2	72.3	74.5	75.4
	Vicuna-7B	14.9	19.9	29.9	35.1	43.2
	Vicuna-13B	23.0	24.8	35.0	43.9	49.2
	Vicuna-33B	43.2	44.1	47.8	53.1	58.5
	Average (accuracy)	35.3	40.8	43.5	47.6 (9.5 % ↑)	53.2 (22.5 % ↑
MATH (4-shot)	Phi-2	10.1	11.6	11.5	13.9	15.4
	Mistral-7B	6.4	10.3	7.9	9.3	10.8
	Llama2-7B	4.1	9.5	6.3	8.6	10.3
	Llama2-7B-Chat	4.6	7.8	6.0	8.0	10.4
	Llama2-13B	4.5	9.7	7.7	9.6	11.4
	Llama2-13B-Chat	5.2	9.8	7.3	9.2	10.5
	Llama3-8B	11.0	16.1	12.8	15.9	16.0
	Llama3-8B-Instruct	18.1	18.6	18.1	18.9	19.7
	Vicuna-7B	3.8	10.1	6.7	8.8	10.5
	Vicuna-13B	3.8	11.1	6.7	9.5	11.0
	Vicuna-33B	6.8	13.1	9.3	11.2	13.4
	Average (accuracy)	7.1	11.6	9.1	11.2 (22.6 % †)	12.7 (39.0 % ↑
HumanEval (0-shot)	Phi-2	45.7	35.1	39.6	45.7	49.4
	Mistral-7B	28.3	32.4	31.2	33.2	35.4
	Llama2-7B	12.2	16.2	14.4	16.2	18.3
	Llama2-7B-Chat	13.4	16.3	12.6	15.6	22.6
	Llama2-13B	17.1	17.7	16.5	19.5	21.3
	Llama2-13B-Chat	19.5	17.2	16.1	17.4	22.0
	Llama3-8B	27.4	23.8	24.6	31.7	37.2
	Llama3-8B-Instruct	56.7	48.1	53.2	56.3	59.5
	CodeLlama-7B	31.1	30.5	28.9	31.2	32.9
	CodeLlama-13B	36.5	39.0	35.4	40.8	40.2
	CodeLlama-34B	48.7	45.1	41.8	49.7	53.2
	Average (pass@1)	30.6		28.6	32.5 (13.7 % †)	35.6 (24.7 % ↑

practical implementation, the Advisor's utility function is defined as the logarithm of the probability of generating the correct answer, while the Receiver's utility function is u(x,c,y)=P(y|x,c), naturally aligning the model's inherent mechanism with the Receiver's behavior. A detailed description and intuition for the setup of the utility function are provided in Appendix A.2.

Datasets. For a comprehensive evaluation of the ability, we select two kinds of tasks: math problems and code generation.

- **GSM8K** [17] is high-quality linguistically diverse grade school math word problems created by human problem writers, which contains 7.5k training problems and 1k test problems.
- MATH [23] is a dataset of challenging competition mathematics problems, which is segmented into 7.5k training problems and 5k testing problems.
- **HumanEval** [12] is a code evaluation benchmark consisting of 164 original programming questions, assessing language comprehension, algorithms, and basic mathematics, with some questions equivalent to simple software interview questions.
- MBPP [4] consists of approximately 1k crowdsourced Python programming problems, covering basic programming knowledge, standard library functionalities, etc. This dataset is only used for the training of the Advisor models.

Advisor and Receiver. In our persuasion framework, we employ two models: an Advisor (small model) and a Receiver (large model). For the Advisor, we select two well-known open-source small models: **GPT-2** [39] (124M) and **Phi-2** [25] (2.7B). To broadly investigate the generalization of the proposed method across various models, we consider several large models as Receivers: **Phi-2** [25] (2.7B), **Llama-2** [46] (7B, 13B), **Llama-3** (8B), **CodeLlama** [41] (7B, 13B, 34B), **Vicuna** [13] (7B, 13B, 33B), and **Mistral** [28] (7B).

Evaluation Metrics. For the math problems, we determine accuracy by extracting the last number from the generated responses and comparing it directly to the ground truth. For the code generation tasks, our evaluation focuses on assessing the functional correctness of LLM-synthesized code. We use the unbiased version of the pass@1 [12] setting for both tasks, namely only generating one result per round. In practice, we use the tool chain-of-thought-hub⁴ and DeepSeek-Coder⁵ to perform the evaluation process for math and code generation tasks, respectively.

Evaluation Settings. For any given input, there is a corresponding set of information item, each item of which is related to the input. In our experiments, we examine four information structures. Given the specific input, the Receiver may observe: (1) *No Information* items, (2) *All Information* items, (3) an item sampled from the *Prior Information* distribution, or (4) an item sampled from the *Posterior Information* distribution. Naturally, the variation in information structure has an impact on the quality of the Receiver's response.

4.2 Results

We evaluate the Advisor with various Receiver models to investigate the effectiveness of its signaling strategy on math problem-solving and code generation tasks. Table 1 shows the Advisor improves the performance of various models through persuasion instead of training. Additional experiments demonstrate that our persuasion framework can identify a non-trivial signaling strategy, which exhibits superior performance in terms of efficiency and generalization.

4.2.1 Performance on Persuasion

To investigate the effectiveness of our persuasion framework, we conduct an experimental evaluation of the Receiver's behavior under prior information distribution and posterior information distribution. Table 1 illustrates that Advisor can significantly improve different Receiver's performance across a variety of tasks. Comparing the Receiver's performance without additional information to that with prior information, we find that additional information enhances the Receiver's performance. From the perspective of persuasion, prior and posterior distributions share the same information set. Instead of training, the Advisor (GPT-2) can significantly enhance the performance of various models, achieving an average improvement of 9.5% on GSM8K, 22.6% on MATH, and 13.7% on HumanEval. When considering the increment in the model parameters for the Advisor, a larger one (Phi-2) enables significant enhancements, with an average improvement of 22.5% on GSM8K, 39.0% on MATH, and a 24.7% increase on HumanEval. One important observation can be noticed: a good signaling strategy by the Advisor can effectively persuade different Receivers.

4.2.2 Impact on Information Structure

In our experiment, we also analyze the impact of different information structures on the Receiver. In the persuasion process, the receiver combines information items from the information set with input to generate a response. From the perspective of prompt engineering, we evaluate the quality of responses when the receiver either disregards information items or considers all information items, to demonstrate the effect of information selection. For 'No Information', it serves as a baseline, equivalent to standard performance testing for LLMs. As shown in Table 1, when the Receiver can access all information items, its performance improves. However, it is noteworthy that for some models, using all information items results in minimal gains or even a decline in performance compared to not using the information. It can be explained that providing too much information disperses the model's attention and risks exceeding the model's maximum window length.

4.2.3 Easy-to-Hard Generalization

In the extended evaluation, we investigate the generalizability of the Advisor. The results presented in Table 1 demonstrate that the Advisor's signaling strategy is effective across various Receiver,

⁴https://github.com/FranxYao/chain-of-thought-hub/tree/main

⁵https://github.com/deepseek-ai/DeepSeek-Coder

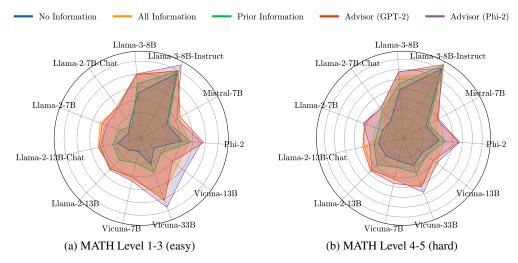


Figure 2: **Performance of Receiver on easy and hard problems.** The Advisor (GPT-2 and Phi-2) is trained on easy problems of training set (MATH level 1-3), and we observe that the capability of the Receiver greatly improved on both easy and hard tasks with the persuasion signal of the Advisor. In both subfigures, our method outperforms scenarios where no information or only prior information is given, and it even surpasses scenarios where all information is provided for most Receiver models.

confirming its broad applicability. Following Sun et al. [44], we evaluate our framework's *Easy-to-Hard Generalization*, which is defined as the ability to address hard tasks by training on simpler ones. We train our Advisor on easy problems (levels 1-3) from the MATH training dataset and assess their effectiveness in persuading various models on both easy (levels 1-3) and complex problems (levels 4-5) of the MATH test dataset. As shown in Figure 2, we observe that advisors not only enhance the performance of various receiver models on easy problems but also improve their performance on hard problems, which are only trained exclusively with supervision on easy problems.

4.2.4 Efficiency on Persuasion

The efficiency of our framework lies in two aspects. On one hand, a well-trained Advisor can persuade various models to elicit better responses. On the other hand, during the inference stage, our method achieves enhanced Receiver's performance with fewer input tokens. To better understand the relationship between performance improvement and prompt length, we design the Average Relative Performance Improvement (ARPI) metric to measure the improvement in performance relative to the increase in prompt length.

Average Relative Performance Improvement (ARPI). To compare the performance of a specific information structure with 'No Information' across several receivers, let R(A) represent the performance of structure A, and let L denote the length of the input prompt tokens. We define ARPI(A|B) as follows:

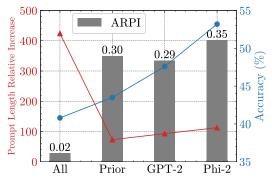


Figure 3: Average Relative Accuracy Improvement on GSM8K. We compare two posterior structure from Advisor with 'All Information' and 'Prior Information'. The left y-axis represents the increase in prompt token length relative to the 'No Information', while the right y-axis displays the average accuracy across various models on GSM8K.

$$ARPI(A|B) = \frac{1}{N} \sum_{i=1}^{N} \frac{R(A) - R(B)}{L(A) - L(B)}.$$
 (7)

ARPI(A|B) presents the relative performance difference of structure A compared with structure B. Figure 3 shows that when the Receiver uses all available information to generate responses, it improves performance relative to using no information, but this results in a 26% increase in the length

of input tokens, thereby increasing the computational cost of inference. In contrast, utilizing our persuasion framework, Phi-2 increases the input token length by only 6.9% while achieving a 22.5% performance improvement, leading to a higher efficiency ratio.

5 Conclusion

In this work, we introduce Bayesian Persuasion Alignment, a novel framework that integrates the concept of Bayesian persuasion with AI alignment. By formulating alignment as a Bayesian Persuasion problem, we employ a smaller model as an Advisor to generate signals that persuade a larger model, the Receiver, to enhance its performance. Our experimental results demonstrate significant improvements in the performance of various large models on mathematical problem-solving tasks and code generation tasks. The theoretical analysis provides an upper bound on the Advisor's regret, highlighting the efficacy of our method in learning the optimal signaling strategy. We hope our approach will inspire future research in integrating information design with alignment, contributing to the development of more efficient and effective AI systems.

Limitations. The effectiveness of persuasion depends on the signaling strategy and is also influenced by the inherent capabilities of the Receiver. If the model itself lacks the ability to complete a certain task, our method may not be effective, which limits the applicability of our framework.

References

- [1] Jacob D Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Conference on Learning Theory (COLT)*, pages 263–274. Citeseer, 2008.
- [2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [3] Itai Arieli and Yakov Babichenko. Private bayesian persuasion. *Journal of Economic Theory*, 182:185–217, 2019.
- [4] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.
- [5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [6] Martino Bernasconi, Matteo Castiglioni, Andrea Celli, Alberto Marchesi, Francesco Trovò, and Nicola Gatti. Optimal rates and efficient algorithms for online bayesian persuasion. In *International Conference on Machine Learning (ICML)*, pages 2164–2183. PMLR, 2023.
- [7] Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.
- [8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 1877–1901, 2020.
- [9] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.
- [10] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. arXiv preprint arXiv:2312.09390, 2023.
- [11] Matteo Castiglioni, Andrea Celli, Alberto Marchesi, and Nicola Gatti. Online bayesian persuasion. Advances in Neural Information Processing Systems (NeurIPS), 33:16188–16198, 2020.

- [12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.
- [13] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.
- [14] Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.
- [15] Paul Christiano, Mark Xu, and Ajeya Cotra. Arc's first technical report: Eliciting latent knowledge. https://www.alignmentforum.org/posts/qHCDysDnvhteW7kRd/arc-s-first-technical-report-eliciting-latent-knowledge, 2021.
- [16] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- [17] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.
- [18] WD Cook and RJ Webster. Caratheodory's theorem. *Canadian Mathematical Bulletin*, 15(2): 293–293, 1972.
- [19] Esin Durmus, Liane Lovitt, Alex Tamkin, Stuart Ritchie, Jack Clark, and Deep Ganguli. Measuring the persuasiveness of language models, 2024. URL https://www.anthropic.com/news/measuring-model-persuasiveness.
- [20] Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. Truthful ai: Developing and governing ai that does not lie. *arXiv preprint arXiv:2110.06674*, 2021.
- [21] Brian J Fogg. Persuasive technology: using computers to change what we think and do. *Ubiquity*, 2002(December):2, 2002.
- [22] Jiarui Gan, Rupak Majumdar, Goran Radanovic, and Adish Singla. Bayesian persuasion in sequential decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 5025–5033, 2022.
- [23] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In Neural Information Processing Systems Track on Datasets and Benchmarks, volume 1, 2021.
- [24] Marius Hobbhahn. Eliciting latent knowledge (elk) distillation/summary. https://www.alignmentforum.org/posts/rxoBY9CMkqDsHt25t/eliciting-latent-knowledge-elk-distillation-summary, 2022.
- [25] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023.

- [26] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. *arXiv* preprint arXiv:2310.19852, 2023.
- [27] Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Juntao Dai, and Yaodong Yang. Aligner: Achieving efficient alignment through weak-to-strong correction. arXiv preprint arXiv:2402.02416, 2024.
- [28] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [29] Emir Kamenica. Bayesian persuasion and information design. *Annual Review of Economics*, 11:249–272, 2019.
- [30] Emir Kamenica and Matthew Gentzkow. Bayesian persuasion. *American Economic Review*, 101(6):2590–2615, 2011.
- [31] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.
- [32] Yuejiang Liu and Alexandre Alahi. Co-supervised learning: Improving weak-to-strong generalization with hierarchical mixture of experts. *arXiv preprint arXiv:2402.15505*, 2024.
- [33] SC Matz, JD Teeny, Sumer S Vaid, H Peters, GM Harari, and M Cerf. The potential of generative ai for personalized persuasion at scale. *Scientific Reports*, 14(1):4692, 2024.
- [34] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.
- [35] OpenAI. Gpt-4 technical report, 2023.
- [36] OpenAI. Introducing superalignment. https://openai.com/blog/introducing-superalignment, 2023. Accessed on July 5, 2023.
- [37] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* (*NeurIPS*), 35:27730–27744, 2022.
- [38] Lorenzo Pacchiardi, Alex J Chan, Sören Mindermann, Ilan Moscovitz, Alexa Y Pan, Yarin Gal, Owain Evans, and Jan Brauner. How to catch an ai liar: Lie detection in black-box Ilms by asking unrelated questions. *arXiv* preprint arXiv:2309.15840, 2023.
- [39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [40] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.
- [41] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2023.
- [42] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. arXiv preprint arXiv:2206.05802, 2022.
- [43] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2023.

- [44] Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. CoRR, abs/2403.09472, 2024.
- [45] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- [46] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [47] Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. Persuasion for good: Towards a personalized persuasive dialogue system for social good. *arXiv preprint arXiv:1906.06725*, 2019.
- [48] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms. *arXiv* preprint arXiv:2401.06373, 2024.
- [49] Zhaowei Zhang, Fengshuo Bai, Mingzhi Wang, Haoyang Ye, Chengdong Ma, and Yaodong Yang. Incentive compatibility for ai alignment in sociotechnical systems: Positions and prospects. *arXiv preprint arXiv:2402.12907*, 2024.

A Experimental Details

In this section, we detail the implementation, including the methodology for constructing information sets, the basic settings for training models, and the prompts utilized in the experiments.

A.1 Information Set Construction

As the dataset itself does not contain information sets, we employ the open-source model Llama3-8B-Instruct ⁽⁶⁾ to rapidly construct the information set for each question within all dataset. Specifically, during model inference, we set the temperature to 0.7 and the top-p to 0.9. Specifically, for mathematical problem-solving tasks, we generate information across seven aspects: Known Data, Objective, Methodology, Procedure Summary, Solution Verification, Assumptions, and Error Analysis. For code generation tasks, the generated aspects include Input and Output, Problem Statement, Test Cases, Logical Deduction, Algorithm Complexity, Error Handling, Edge Cases, and Code Structure. The entire process of information set construction is automated, with the decision on which aspects to generate also being determined by the LLM. For the prior distribution of information items, we calculate the conditional probabilities for each information item given a problem, generated by the model, and normalize these probabilities to form a valid distribution. We provide the complete prompts used for data generation in Appendix A.4.

A specific example

<QUESTION>

On Thursday the Meat Market sold 210kg of ground beef. On Friday they sold twice that amount. On Saturday they only sold 130kg. On Sunday they sold half of what they sold Saturday. If they originally planned to sell only 500kg, how much meat did they sell beyond their original plans?

<EXTRA INFORMATION>

"Known Data": Thursday: 210kg of ground beef sold; Friday: Twice the amount sold on Thursday, which is 2 x 210kg; Saturday: 130kg of ground beef sold; Sunday: Half of what was sold on Saturday, which is 0.5 x 130kg; Original plan: 500kg
"Objective": To calculate how much meat was sold beyond the

original plan
"Methodology": 1. Calculate the total amount of ground beef sold on Thursday, Friday, Saturday, and Sunday. 2. Calculate the total amount of ground beef sold beyond the original plan "Procedure Summary": 1. Add the amount sold on Thursday, Friday, Saturday, and Sunday: 210 + 420 + 130 + 65 2. Subtract the original plan from the total amount sold "Solution Verification": 1. Check if the total amount sold is equal to the sum of the amounts sold on each day. 2. Check if the answer makes sense in the context of the problem

"Assumptions": The data provided is accurate and complete. The calculations are performed correctly "Error Analysis": Potential errors may occur due to incorrect

calculation or misinterpretation of the data. Double-checking the calculations and verifying the answer against the given data can help identify and correct any errors

A.2 Utility Function

In our persuasion, the Receiver's utility function u(x, c, y) is continuous and dependent on its response $y \in \mathcal{Y}$ to the input $x \in \mathcal{X}$ and the associated information item $c \in \mathcal{C}_x$. Similarly, the Advisor has a continuous utility function v(x, c, y), which is contingent on the Receiver's response, input, and

⁽⁶⁾ https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

associated information item. In practical implementation, the utility function of the Advisor is defined as the logarithm of the probability of generating the correct answer, given the input x and the information item c. For the utility function u(x,c,y), a natural idea is to set the conditional probability P(y|x,c) as utility. Autoregressive language model generate responses by continuously choosing the next token with the highest probability, ultimately producing the response with the maximum probability. This behavior aligns precisely with the Receiver selecting the optimal response based on the maximum utility. Therefore, in the experiment, given the input x and information item c, the response generated by the Receiver is equivalent to selecting a response from the set (1).

A.3 Training Hyperparameters

In our experiments, we train two models **GPT-2** [39] (124M) and **Phi-2** [25] (2.7B), utilizing the training datasets from GSM8K and MATH for mathematical problem-solving tasks, and MBPP for code generation tasks. Throughout the training, we employe the AdamW optimizer with hyperparameters set to $\beta_1=0.9,\,\beta_2=0.95,$ and $\epsilon=10^{-5}.$ Additionally, We use a cosine learning rate schedule with a maximum learning rate of $5\times10^{-5}.$ All models are trained on 4 NVIDIA A800 GPUs.

A.4 Prompts

In our experiment, we employ distinct prompts at different stages, which included the construction of the information set for math problems and code generation tasks, the generation of signals by the Advisor, and the generation of responses by the Receiver. Here, we present the prompts used throughout our experiment.

A.4.1 Math Tasks

Prompt for the construction of information sets on math problems

<INSTRUCTION>

Please provide key information on the following aspects:

- 1. Known Data: List all numerical values and conditions given in the problem.
- 2. Objective: Clearly define the specific calculation or problem that needs to be solved.
- 3. Methodology: Describe the mathematical formulas or logical reasoning required to solve the problem.
- 4. Procedure Summary: Outline the solution steps from the given data to the resolution of the problem.
- 5. Solution Verification: Suggest methods to verify the correctness of the answer.
- 6. Assumptions: List any assumptions made to simplify the problem or calculation.
- 7. Error Analysis: Identify potential errors or mistakes that may occur during the calculation.

Ensure that the information provided is accurate, precise to facilitate the correct solution.

```
<QUESTION>
```

A.4.2 Response Generation

Prompt for Receiver generate a response

```
<QUESTION>
```

•••

<EXTRA INFORMATION>

Integrate with the additional context to form a thorough and insightful answer. {information item}

<ANSWER>

Let's think step by step.

A.4.3 Code Generation Tasks

Prompt for the construction of information sets on code task

<INSTRUCTION>

Please provide key information on the following aspects:

- 1. Input and Output: Clearly specify the function's parameters and return types.
- 2. Problem Statement: Understand the problem to be solved and the expected solution.
- 3. Test Cases: Design test cases based on edge cases and special situations.
- 4. Logical Deduction: Determine the basic logic for solving the problem based on the description and examples.
- 5. Algorithm Complexity: Evaluate the time and space complexity of the designed algorithm.
- 6. Error Handling: Consider handling potential errors and exceptions.
- 7. Edge Cases: Identify extreme cases in the problem.

Ensure that the information provided is accurate, precise to facilitate the correct solution.

```
<QUESTION>
```

A.4.4 Signal Generation

Prompt for Advisor generate a signal

<INSTRUCTION>

Summarize below information and present the most important details in an accurate and precise format

```
<EXTRA INFORMATION>
```

```
{all information items}
```

A.5 The Training Code of Persuasion

The pseudocode below shows the basic training process of our Bayesian persuasion framework.

```
def compute_loss(advisor, inputs):
      # Step1. smaple a information item
      sample_infos = ...
      # Step2. produce signals by Advisor, condition on sample_infos
      signals_ids = advisor.generate(max_new_tokens=50,)
      # Step3. update belief on info by bayes rule
      posterior_belief = update_posterior_belief(prior, info_items,
      signals_ids)
      # Step4. Receiver choose best response
      item_index = torch.max(posterior_belief, dim=1)
9
      advisor_utility = get_advisor_utility(receiver, inputs_ids,
11
      sample_infos)
      # Step5. calculate the loss of Advisor
      loss = (posterior_belief * advisor_utility).mean()
13
      return loss
14
```

B Additional Experiments

B.1 Generalization on various tasks

We investigate the generalization of the signaling strategy across different Receivers in Section 4.2.1 and across varying levels of difficulty in Section 4.2.3. In this section, we evaluate the generalization of our well-trained signaling strategy across various tasks. Specifically, we train the Advisor using the GSM8K training dataset and assess its performance on the MATH dataset. Concurrently, we train the Advisor using the MATH training dataset and assess its performance on GSM8K. As shown in

Table 2: **Performance of various Receivers under persuasion.** We report the accuracy on GSM8K and MATH. "Posterior Information" refers to sampling the information item from the posterior distribution, influenced by the Advisor. The Advisor for math tasks differs from that for code generation tasks. Arrows indicate performance improvements relative to the prior distribution.

Task	Receiver	No Information	All Information	Prior Information	Posterior Information	
					Advisor (GPT-2)	Advisor (Phi-2)
GSM8K (8-shot)	Phi-2	56.0	41.0	56.8	57.3	59.3
	Mistral-7B	34.3	48.0	45.7	47.3	51.3
	Llama2-7B	15.1	36.6	27.2	33.0	44.0
	Llama2-7B-Chat	21.8	31.8	37.3	40.3	48.3
	Llama2-13B	25.2	38.9	36.2	41.7	43.7
	Llama2-13B-Chat	33.9	37.3	36.1	37.7	37.7
	Llama3-8B	47.6	54.0	53.7	54.4	54.4
	Llama3-8B-Instruct	73.5	72.2	72.3	74.3	74.3
	Vicuna-7B	14.9	19.9	29.9	32.6	39.6
	Vicuna-13B	23.0	24.8	35.0	38.3	45.3
	Vicuna-33B	43.2	44.1	47.8	50.8	55.8
	Average (accuracy)	35.3	40.8	43.5	46.2 (6.2 % ↑)	50.3 (15.6% †)
MATH (4-shot)	Phi-2	10.1	11.6	11.5	13.8	14.8
	Mistral-7B	6.4	10.3	7.9	9.1	10.8
	Llama2-7B	4.1	9.5	6.3	8.5	10.3
	Llama2-7B-Chat	4.6	7.8	6.0	7.9	9.3
	Llama2-13B	4.5	9.7	7.7	9.3	10.5
	Llama2-13B-Chat	5.2	9.8	7.3	8.8	10.4
	Llama3-8B	11.0	16.1	12.8	15.5	16.0
	Llama3-8B-Instruct	18.1	18.6	18.1	18.8	19.3
	Vicuna-7B	3.8	10.1	6.7	8.7	11.1
	Vicuna-13B	3.8	11.1	6.7	9.1	13.0
	Vicuna-33B	6.8	13.1	9.3	10.6	12.2
	Average (accuracy)	7.1	11.6	9.1	10.9 (19.8% †)	12.5 (37.3 % †)

Table 2, the Advisor is capable of enhancing the performance of all Receivers on both GSM8K and MATH to varying degrees. When GPT-2 acts as the Advisor, it facilitates performance improvements for multiple Receivers, with an average performance increase of 6.2% on GSM8K and 19.8% on MATH. In contrast, Phi-2 achieves more notable performance enhancements, with gains of 15.6% on GSM8K and 37.3% on MATH.

C Proof of Theorem 1

Assumption 1. The prior distribution μ^0 is in the interior of $\Delta(\mathcal{C})$, i.e., $\mu_c^0 > 0$ for all $c \in \mathcal{C}$.

Definition 1 (Linear Map). A vector-valued function $f: X \to \mathbb{R}^D$ is said to be linear if there exists a matrix $M \in \mathbb{R}^{D \times M}$ such that f(x) = Mx for all $x \in X \subseteq \mathbb{R}^M$.

Theorem 2 (Caratheodory's Theorem [18]). Let $S \subseteq \mathbb{R}^D$ be a set. Then, any point in the convex hull of S can be expressed as a convex combination of at most D+1 points from S.

Theorem 3 ([6], Theorem 3.2). If X is a polytope and f is a linear map, then there exist algorithms implementing the Carathéodory decomposition and the inverse map f^{\dagger} .

Proof. If X is a polytope and f is a linear map, then f is also a polytope. By Caratheodory's theorem, every point in f(X) can be expressed as a convex combination of at most D+1 vertices of f(X), where D is the dimension of f(X).

Given any $z \in f(X)$, the Carathéodory decomposition algorithm finds at most D+1 vertices $\{z_1,\ldots,z_{D+1}\}\subseteq f(X)$ and convex coefficients $\{\lambda_1,\ldots,\lambda_{D+1}\}$ such that $z=\sum_{i=1}^{D+1}\lambda_iz_i$. This can be done by solving a linear program.

For the inverse map, given any $z \in f(X)$, we want to find an $x \in X$ such that f(x) = z. Since f is linear, we have f(x) = Mx for some matrix M. Finding x is thus equivalent to solving the linear system Mx = z. Since $z \in f(X)$, this system is guaranteed to have a solution, which can be found using Gaussian elimination. \Box

Corollary 1 ([6], Corollary 3.4). *Under the assumptions of Theorem 3, there exists a regret minimizer* R *such that Algorithm 1 guarantees cumulative regret*

$$R_T \le 16D^{3/2}\sqrt{T\log T},\tag{8}$$

where D is the dimension of f(X).

Proof. To obtain the regret bound, we equip Algorithm 1 with a suitably-defined regret minimizer \mathcal{R} . In particular, \mathcal{R} works by observing the realized utility $v(y_t, c_t)$, since the sender does not directly play ϕ_t , but rather draws an action y_t according to ϕ_{t,c_t} . Such a regret minimizer \mathcal{R} can be implemented by the algorithm introduced by [1], as any polytope in \mathbb{R}^D has a D-self concordant barrier [34] (Theorem 2.5.1). This yields the stated regret bound [1] (Theorem 1).

With the above theorems and corollary, we are now ready to prove Thereom 1.

Proof. The proof of this theorem will proceed in three steps.

Step 1: Show that the set of direct and persuasive signaling schemes \mathcal{P} is a polytope.

To see this, note that \mathcal{P} can be described by the following linear constraints:

$$\sum_{y \in \mathcal{Y}} \phi_c(y) = 1, \forall c \in \mathcal{C}, \tag{9}$$

$$\sum_{c \in \mathcal{C}} \mu_c \phi_{\mathcal{C}}(y)(v(y, c) - v(y', c)) \ge 0, \forall y, y' \in \mathcal{Y}, \tag{10}$$

$$\phi_c(y) > 0, \forall c \in \mathcal{C}, y \in \mathcal{Y}.$$
 (11)

Constraint equation 9 ensures that ϕ_c is a valid probability distribution for each c. Constraint equation 10 is the persuasiveness constraint. Constraint equation 11 ensures non-negativity. As these are all linear constraints, \mathcal{P} is a polytope.

Step 2: Define a linear map $f: \mathcal{P} \to \mathbb{R}^m$.

Let $f: \mathcal{Y} \to \mathbb{R}^m$ be defined as $f(\phi) = [-v(\phi,y)]_{y \in \mathcal{Y}}$ for all $\phi \in \mathcal{P}$, where $v(\phi,y) = \sum_{c \in \mathcal{C}} \mu_c \phi_c(y) v(y,c)$ is the Advisor's expected utility for action y under signaling scheme ϕ . We can verify that f is linear:

$$f(\alpha\phi_1 + \beta\phi_2) = [-v(\alpha\phi_1 + \beta\phi_2, y)]_{y \in \mathcal{Y}}$$

$$= [-\alpha v(\phi_1, y) - \beta v(\phi_2, y)]_{y \in \mathcal{Y}}$$

$$= \alpha [-v(\phi_1, y)]_{y \in \mathcal{Y}} + \beta [-v(\phi_2, y)]_{y \in \mathcal{Y}}$$

$$= \alpha f(\phi_1) + \beta f(\phi_2)$$

for any $\phi_1, \phi_2 \in \mathcal{P}$ and $\alpha, \beta \in \mathbb{R}$.

Step 3: Apply Corollary 1 to derive the regret bound.

Since f is a linear map from \mathcal{P} to \mathbb{R}^m , we have $f(\mathcal{P}) \subseteq \mathbb{R}^m$, so the dimension of $f(\mathcal{P})$ is at most m. By Corollary 1, if the dimension of $f(\mathcal{P})$ is D, then there exists a regret minimizer with regret bound

$$R_T \le 16D^{3/2} \sqrt{T \log T}.$$

In our setting, $D \leq m$. Therefore, we get the following regret bound:

$$R_T = O(m^{3/2} \sqrt{T \log T}).$$

Putting everything together, we have shown that the regret of Algorithm 1 is upper bounded by $O(m^{3/2}\sqrt{T\log T})$, where $m=|\mathcal{Y}|$ is the size of the Receiver's response space.