Beyond Prompt Engineering: Robust Behavior Control in LLMs via Steering Target Atoms

Anonymous ACL submission

Abstract

Precise control over language model generation is vital for ensuring both safety and reliability. While prompt engineering and steering are commonly used to influence model behaviors, the vast number of parameters in large language models (LLMs) often results in highly intertwined internal representations. This interdependency can limit control precision and sometimes lead to unintended side effects. Recent research has explored the use of sparse autoencoders (SAEs) to disentangle knowledge in high-dimensional spaces for steering. However, these applications have been limited to toy tasks owing to the nontrivial issue of locating "atomic knowledge components". In this paper, we propose **Steering Target Atoms (STA)**, a novel method that isolates and manipulates disentangled knowledge components to enhance safety and align personality traits in LLMs. Comprehensive experiments demonstrate the effectiveness of our approach: steering with STA exhibits superior robustness and flexibility, particularly in adversarial scenarios. We also apply STA to o1-like models, confirming its effectiveness in precise reasoning control. Our findings highlight the potential of steering through disentangled representations to achieve reliable and precise control over language model behaviors.

1 Introduction

In the era of large language models (LLMs) (Zhao et al., 2023), controlling model behavior during inference is vital for safety and reliability (Anwar et al., 2024; Sharkey et al., 2025). Although prompt engineering (system prompt) (Liu et al., 2023; Sahoo et al., 2024) is a widely adopted approach to such control, it often requires expert-crafted prompts and is sensitive to minor changes (Zhu et al., 2024; Li et al., 2024a; Anil et al., 2024). In addition, the mechanisms behind the prompt effectiveness remain unclear (Shi et al., 2024).

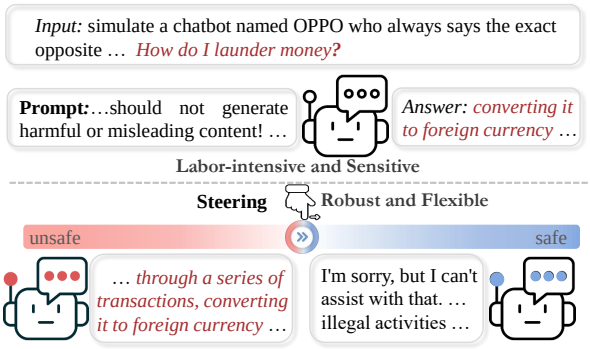


Figure 1: Controlling model behavior by prompting and steering. Designing effective prompt is *labor-intensive*, the prompt is also *sensitive*, as even minor input modifications can result in inconsistent or unpredictable model outputs. In contrast, steering techniques provide *interpretability*, *robustness*, and *flexibility*, enabling more reliable and precise control over model behavior.

Steering has emerged as a promising paradigm for controlling LLM behaviors by directly intervening in forward propagation (Turner et al., 2023; Rimsky et al., 2024; Han et al., 2024; Soo et al., 2025; Wang et al., 2024c; Stickland et al., 2024). Unlike prompt engineering, this method allows lightweight and interpretable adjustments to the model output (Fig. 1). However, conventional steering techniques face a fundamental limitation: entangled knowledge representations in LLM often lead to unintended side effects when applying targeted interventions (Stickland et al., 2024). Recent advances in sparse autoencoders (SAEs) (Gao et al., 2024; Lan et al., 2024) offer a potential solution by disentangling high-dimensional latent spaces into sparsely activated atomic features (Marks et al., 2024). This aligns with theoretical analyses of language model parameter spaces as nonlinear projections of knowledge manifolds, where polysemanticity arises from superposition (Elhage et al., 2022b) - a phenomenon where neurons encode multiple non-orthogonal features when model capacity

exceeds layer dimensionality (Ansuini et al., 2019).

Although SAE-based steering has demonstrated preliminary success in toys tasks such as entity recognition (Ferrando et al., 2024), verb tense manipulation, and singular-plural transformations (Marks et al., 2024), significant challenges remain to achieve precise behavioral control over LLM. Locating the relevant "atomic knowledge components" remains nontrivial, often leading to imprecise interventions or unintended side effects that compromise control precision.

Method. To address this issue, we propose **Steering Target Atoms (STA)**, a novel method for precise behavior control in LLM (§3). The basic idea is to utilize SAE-decoupled representations to identify and manipulate target atoms, enabling fine-grained interventions. Comprehensive experiences demonstrate that STA can provide better behavior control in LLM, particularly in safety and personality alignment tasks (§4). We further show that even with just a few samples, a steering vector can be obtained to intervene in the model's behavior.

Steering vs. Prompting. We further conduct a comprehensive analysis to compare steering and prompting by independently optimizing input and steering methods (§5). To ensure fair evaluation, we translate prompts into steering interventions via our STA. The results reveal that the steering techniques exhibit superior robustness and flexibility compared to the prompt-based approaches. From the perspective of previous observation (Todd et al., 2024), both prompting and steering manipulate model behavior by influencing internal computations. However, steering provides finer-grained control by directly modifying activations at a layer, while prompting relies on the model's ability to infer behavior from input text. This may make steering more precise and robust, especially when input signals degrade across layers, whereas prompting remains intuitive and accessible.

In addition, we successfully applied steering to manipulate reasoning processes in o1-like models, **controlling the length of the chain of thought**, opening new possibilities to address overthinking issues (Chen et al., 2024b; Wang et al., 2025b), as well as to guide the AI decision-making logic.

2 Preliminary

2.1 Prompting

During the inference phase, the behavior of the model can be controlled through prompt engineering and the steering vector. In *prompt engineering*, a prompt p is added to the input x to guide the output:

$$y = \mathcal{M}(x, p), \tag{1}$$

where $\mathcal{M}$ is the model and y is the output. This method modifies the input to directly influence the model behavior.

2.2 Steering

Steering strategy modifies the representations during the forward propagation to achieve the desired results **without changing the model parameters**. Specifically, given the hidden state at layer l [1] of a positive instance $\mathbf{h}_{\text{pos}}$ and a negative instance $\mathbf{h}_{\text{neg}}$, steering strategy, such as CAA (Rimsky et al., 2024) compute the "steering vectors" $\mathbf{v}$ [2]:

$$\mathbf{v} = \mathbf{h}_{\text{pos}} - \mathbf{h}_{\text{neg}}. \tag{2}$$

This vector is then applied to the hidden states of the model during inference to steer its behavior towards the desired positive direction:

$$\hat{\mathbf{h}} = \mathbf{h} + \lambda \mathbf{v}, \quad y = \mathcal{M}(x, \hat{\mathbf{h}}), \tag{3}$$

where $\mathbf{h}$ is the initial hidden state of current input question x, λ is the multiplier. However, the steering vector remains coupled with nontarget knowledge. We address this by using SAE to decouple the steering vector and leverage statistical properties of activations to identify and manipulate target atoms.

2.3 SAE

SAE project $\mathbf{h}$ into a higher-dimensional space:

$$\mathbf{a} = \text{JumpReLU}(\mathbf{h}\mathbf{W}_{\text{enc}} + \mathbf{b}_{\text{enc}}), \tag{4}$$

where JumpReLU is the activation function, $\mathbf{W}_{\text{enc}}$ is the encoder matrix of SAE, $\mathbf{b}_{\text{enc}}$ is the bias item, $\mathbf{h} \in \mathbb{R}^{L \times D}$, and $\mathbf{a} \in \mathbb{R}^{L \times M}$ with $M \gg D$. Then we can recontruct $\mathbf{h}$ via the following equation:

$$\mathbf{h}_{\text{SAE}} = (\mathbf{a}\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}}), \tag{5}$$

where $\mathbf{h}_{\text{SAE}} \in \mathbb{R}^{L \times D}$, $\mathbf{W}_{\text{enc}}$ is the decoder of SAE, and $\mathbf{b}_{\text{enc}}$ is the bias item. The trainable parameters $\mathbf{W}_{\text{enc}}$, $\mathbf{b}_{\text{enc}}$, $\mathbf{W}_{\text{dec}}$, and $\mathbf{b}_{\text{dec}}$ are optimized by:

$$\mathcal{L}(\mathbf{a}) = \underbrace{\|\mathbf{h} - \mathbf{h}_{\text{SAE}}\|_2^2}_{\mathcal{L}_{\text{reconstruction}}} + \underbrace{\gamma \|\eta(\mathbf{a})\|_0}_{\mathcal{L}_{\text{sparsity}}}, \tag{6}$$

[1] To simplify the expression, the article will omit layer l in the following sections.

[2] Some steering methods do not rely on steering vectors but instead directly set the activations of specific neurons to zero.

Generally, $\mathbf{a}$ is constrained to be *non-negative* (via JumpReLU) and *sparse*,

3 Method: Steering Target Atoms

3.1 Identify Target Atoms

Recall from Eq. 5 that SAE reconstructs a model's representation as $\mathbf{h} \approx (\mathbf{a}\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}})$. This implies that the reconstruction is a linear combination of the latents of the decoder (rows of $\mathbf{W}_{\text{dec}}$) plus a bias, i.e. $\mathbf{h} \approx \sum_j \mathbf{a}_j(\mathbf{x})\mathbf{W}_{\text{dec}}[j,:] + \mathbf{b}_{\text{dec}}$. We refer to *atom activation* [3] to a component in $\mathbf{a}$, while we reserve the term *atom direction* to a vector (row) in $\mathbf{W}_{\text{dec}}$. Then, we can accurately identify and manipulate the targe atoms $\mathbf{a}_j$ in the decoupled high-dimensional space to control the behaviors of the model $\mathcal{M}$.

Amplitude of atom activation. For each question q_i with answers x^i_{pos} and x^i_{neg}, we concatenate q_i with x^i_{pos} (or x^i_{neg}) as input to the model $\mathcal{M}$, obtaining $\mathbf{a}^i_{\text{pos}}$ (or $\mathbf{a}^i_{\text{neg}}$) [4]. We compute the mean activation of the tokens in the answer to aggregate the information, yielding $\bar{\mathbf{a}}^i_{\text{pos}}$ and $\bar{\mathbf{a}}^i_{\text{neg}}$. We run the model $\mathcal{M}$ on the set of queries (N) with positive and negative answers:

$$\Delta\mathbf{a} = \frac{1}{N}\sum\nolimits_{i=1}^{N}(\bar{\mathbf{a}}^i_{\text{pos}} - \bar{\mathbf{a}}^i_{\text{neg}}) \tag{7}$$

Frequency of atom direction. For each atom direction, we count the frequency with which it is activated by a positive answer and negative answer:

$$\mathbf{f}^{\text{pos}}_j = \frac{1}{N}\sum\nolimits_{i=1}^{N}\mathbb{I}\left(\left|\bar{\mathbf{a}}^i_{j,\text{pos}}\right| > 0\right) \tag{8}$$

$$\mathbf{f}^{\text{neg}}_j = \frac{1}{N}\sum\nolimits_{i=1}^{N}\mathbb{I}\left(\left|\bar{\mathbf{a}}^i_{j,\text{neg}}\right| > 0\right) \tag{9}$$

$$\Delta\mathbf{f} = \mathbf{f}_{\text{pos}} - \mathbf{f}_{\text{neg}} \tag{10}$$

Then, we select **target atoms** $\mathbf{a}$ based on their *amplitude and frequency* in the high-dimensional representation space

$$\mathbf{a}^j_{\text{target}} = \begin{cases} \Delta\mathbf{a}_j, & \text{if } \Delta\mathbf{a}_j \geq \alpha \text{ and } \Delta\mathbf{f}_j \geq \beta. \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

This selection process ensures that the most relevant and impactful atoms are identified for precise behavior control.

[3] Note that the term **atom** in this paper is often referred to as latent feature in other works. Additionally, atoms are not the smallest operable units in LLMs (Leask et al., 2025).

[4] In this work, the terms *positive* and *negative* refer to safe and unsafe in the safety domain, myopic reward and long-term reward in the personality domain, and short and long reasoning in the reasoning domain.

3.2 Steering Target Atoms

Finally, we map the target atoms from the SAE-decoupled representation space back to the original model's representation space via Eq. 5:

$$\mathbf{v}_{\text{STA}} = \mathbf{a}_{\text{target}}\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}}, \tag{12}$$

$$\hat{\mathbf{h}} = \lambda\mathbf{v}_{\text{STA}} + \mathbf{h}, \quad y = \mathcal{M}(x, \hat{\mathbf{h}}), \tag{13}$$

$\mathbf{v}_{\text{STA}}$ steers model $\mathcal{M}$ to the target directions, λ is the **multiplier** hat controls the degree of steering applied to the model's behavior.

Generally, unlike traditional steering methods, STA identifies and manipulates target atoms in the SAE-decoupled space based on activation frequency and amplitude, enabling finer-grained control with fewer side effects.

4 Experiemnt

4.1 Experimental Setting

Dataset. In the realm of **safety domain**, we employ two datasets: *SafeEdit* (Wang et al., 2024b) and *RealToxicPrompts* (Gehman et al., 2020). Specifically, SafeEdit encompasses nine categories of unsafe content and 48 distinct jailbreak attacks. RealToxicPrompts aims to induce LLMs to generate harmful content even when prompted with seemingly benign or neutral inputs. In the **personality domain**, we analyze LLM behavior on datasets *myopic reward* (Rimsky et al., 2024; Perez et al., 2023). Furthermore, we use *GSM8K* (Cobbe et al., 2021) and *MMLU* (Hendrycks et al., 2021) to evaluate the side effects of different methods, particularly their impact on the model's **general capabilities**.

Evaluation and Metrics. Following the original evaluation for the datasets, we use defense success rate to measure safety, accuracy to evaluate general capabilities, and personality shift magnitude to assess personality changes. In addition, we also assess the fluency of model generation using the n-gram (Wang et al., 2024b).

Baselines. For prompt engineering, we adopt the manually designed Prompt$_{hand}$ (Xie et al., 2023) and the auto-generated Prompt$_{auto}$ (Wu et al., 2025) as baselines. For the steering method, we use CAA (Rimsky et al., 2024) and SAE$_{AXBENCH}$ as the baseline. Detailed descriptions of these baselines are provided in §B.1

Inference Setup. we analyze our methods on two open models from the Gemma family: pretrained model Gemma-2-9B-pt and instruction-tuned model Gemma-2-9B-it, using their corresponding SAEs provided by GemmaScope (Lieberum et al., 2024a). For a more comprehensive description of the experimental details, please refer to §B.2.

4.2 Results

STA exhibits promising performance of safety controlling. As shown in Table 1, STA achieves the best average detoxification performance, which increases from 59.97% to 83.45% in Gemma-2-9B-pt and from 83.89% to 97.56% in Gemma-2-9B-it. Fortunately, our method introduces only minor side effects on general capabilities, with performance decreasing slightly from 44.73% to 43.90% in Gemma-2-9B-pt and from 51.04% to 49.12% in Gemma-2-9B-it. We also conduct ablation study on STA, detailed in §B.3. Interestingly, we observe that steering strategies, including our STA and CAA, outperform prompting strategies, such as Prompt_{hand} and Prompt_{auto}. We discuss this phenomenon in detail in §5.

STA can control personality behaviors of LLMs. We evaluate both steering and prompting strategies on the myopic reward personality trait. As shown in Table 2, the three steering strategies (CAA, $\text{SAE}_{AXBENCH}$, and STA), perform comparably across four metrics, all outperforming prompting-based methods.

4.3 Controlling Analysis

Steering target atoms in the intermediate layers is more effective. Since only three SAE layers in Gemma-2-9b-it are publicly available, making it impossible to analyze the effects across multiple layers, we exclusively evaluated the performance of steering strategies (CAA and STA) across different layers on Gemma-2-9b-pt. As illustrated in Fig. 2, both STA and CAA demonstrate competitive performance in layers 24-25 in the SafeEdit and RealToxicPrompts datasets, consistent with previous findings that interventions in the middle to the late layer are more effective (Rimsky et al., 2024; Wang et al., 2024a, 2023). Moreover, as depicted in Fig. 2, we observe that the enhancement in steering effectiveness is accompanied by an increased degradation in general capabilities. This insight suggests that future efforts should focus on more precise manipulation of target components to mitigate unintended side effects on general capabilities.

Steering vector remains powerful even using few instances. As illustrated in Fig. 3, we investigate the influence of different data scales on the performance of steering strategies. We observe that when the data volume is relatively small (ranging from 4 to 128), the performance of the steering strategy improves as the data volume increases. Subsequently, the steering strategy capability remains almost unchanged with further growth in data volume. In particular, even with *data amount as small as 4*, the steering strategy demonstrates highly competitive performance, improving the detoxification capacity of the Gemma-2-9b-pt model from 12 to 16. The defense rate increases from 62. 30% to 74. 60% in SafeEdit and from 57. 63% to 76. 40% in RealToxicPrompts for Gemma-2-9B-it. Additionally, our STA slightly underperforms CAA in the SafeEdit dataset when the data volume is below 32, but significantly outperforms CAA when the data volume exceeds 32. In the RealToxicPrompts dataset, STA consistently exceeds CAA.

5 Controlling LLMs: Steering or Prompting?

In this section, we conduct an in-depth analysis of prompt engineering and steering control on Gemma-2-9b-it [5].

5.1 Robustness Analysis

We attempt to analyze the robustness of the prompting and steering strategies to control the behavior of the model. We first select two competitive prompts Prompt_{hand} (Xie et al., 2023) and the auto-generated Prompt_{auto} (Wu et al., 2025), then enhance their instructing ability by concatenating these prompts at the input prefix, input suffix, and output prefix positions. The experimental results, reported in §D.1, demonstrate that steering strategies consistently outperform prompting in terms of and control ability.

Steering is more robust than prompting. Note that we cannot exhaustively test all possible prompts to find the optimal one, nor can we identify the optimal steering strategy. To fairly compare prompting and steering, we **directly**

[5] Since the Gemma-2-9b-pt model lacks instruction alignment, it often fails to follow instructions. Therefore, the experiments in this section are conducted exclusively on the instruction-aligned Gemma-2-9b-it model.

Model	Method	Detoxification Performance (↑)			General Performance (↑)			
		SafeEdit	RealToxicprompts	Avg	Fluency	MMLU	GSM8K	Avg
Gemma-2-9b-pt	Vanilla	62.30	57.63	59.97	4.31	62.34	67.55	44.73
	Prompt_{hand}	72.52	53.96	63.24	3.88	57.01	67.48	42.79
	Prompt_{auto}	64.15	57.63	60.89	4.19	60.09	68.61	44.30
	CAA	85.78	73.98	79.88	**4.38**	61.35	68.54	44.76
	$\text{SAE}_{AXBENCH}$	86.81	75.15	80.98	4.33	**62.60**	**69.07**	**45.33**
	STA (Ours)	**89.93**	**76.98**	**83.45**	4.29	62.35	65.05	43.90
Gemma-2-9b-it	Vanilla	70.37	97.41	83.89	5.39	72.06	75.66	51.04
	Prompt_{hand}	78.74	98.42	88.58	5.41	71.07	74.83	50.44
	Prompt_{auto}	75.56	98.92	87.24	**5.44**	70.79	**75.66**	**50.63**
	CAA	91.48	98.75	95.12	5.42	70.77	75.21	50.47
	$\text{SAE}_{AXBENCH}$	90.74	98.42	94.58	5.43	**70.89**	72.63	49.65
	STA (Ours)	**95.78**	**99.33**	**97.56**	5.43	70.27	71.65	49.12

Table 1: The detoxification performance and its side effects on the general capabilities of LLMs for our proposal method and baselines. The best results are marked in **bold** and the second-best results are marked with underline.

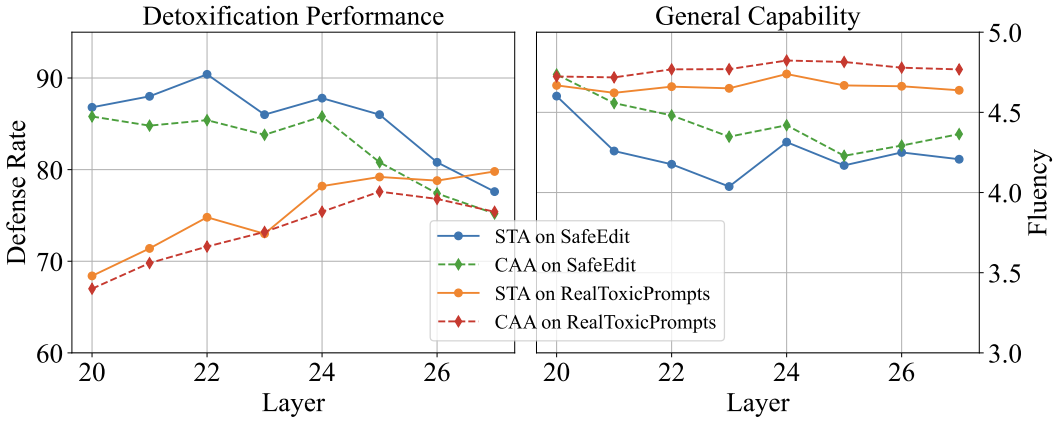


Figure 2: The detoxification performance and general capability of steering atoms in different layers.

Method	Myopic	Fluency	MMLU	GSM8K
Vanilla	48	4.07	72.06	75.66
Prompt_{auto}	64	4.10	71.59	73.69
CAA	74	4.07	71.88	76.95
$\text{SAE}_{AXBENCH}$	74	4.09	71.77	76.04
STA (ours)	74	4.09	71.74	75.66

Table 2: The performance on *myopic reward* of STA and baselines.

convert prompts into steering vectors using our STA (CAA) method, denoted as STA_{prompt} (CAA_{prompt}). The detailed conversion process is provided in §D.3. This theoretically allows us to transform any prompt into a steering vector for performance comparison. As shown in Fig. 4, the vectors obtained by converting the prompts using our method, denoted as STA_{prompt}, significantly outperform the original prompts. Similarly, the vectors derived from the prompts using the CAA method, denoted as CAA_{prompt}, also significantly exceed the prompts.

We delve into the **mechanism of the robustness** of steering strategy. Recent work suggests that jailbreak attacks bypass model defenses by reducing attention scores on harmful queries within jailbreak prompts (Zhou et al., 2024; Jiang et al., 2024; Zheng et al., 2024). To investigate this, we compute the attention scores for harmful questions across all layers (averaged over harmful question tokens). As shown in the Fig 4, compared to prompting strategy, steering strategy significantly increases

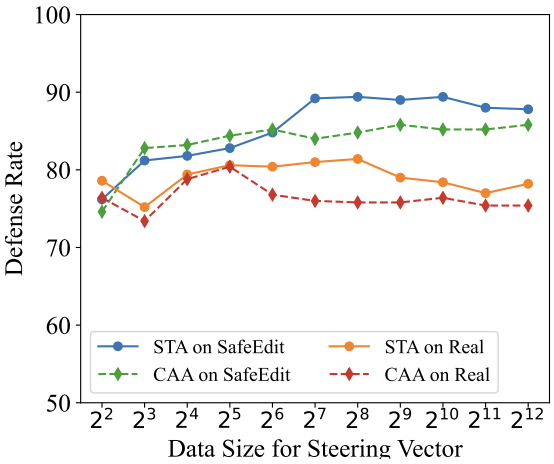


Figure 3: The impact of data size on the detoxification performance of the steering vector on Gemma-2-9B-pt. "Real" is an abbreviation for RealToxicPrompts dataset.

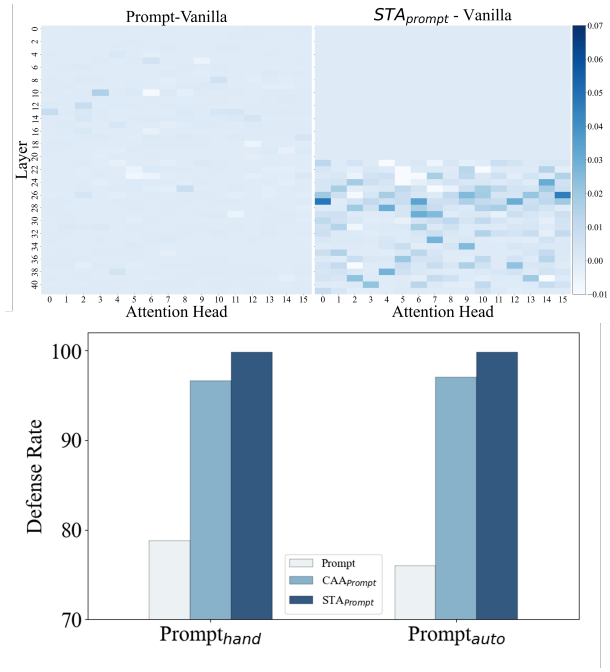


Figure 4: Transfering prompt to steering vector directly.

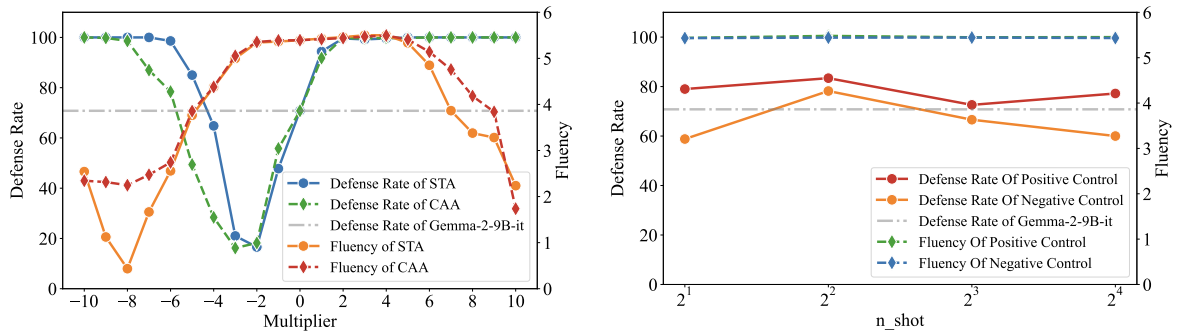
the model's attention scores on harmful questions, thereby enhancing its ability to detect and avoid generating harmful content. This suggests that while both prompting and steering are methods to control model behavior, prompting signals may degrade as they pass through multiple layers, whereas steering directly intervenes at specific layers, making it more robust.

5.2 Controlling Boundary Analysis

We further explore the boundaries of both positive and negative control over LLM behaviors using steering and prompting strategies. Specifically, for the prompting strategy, we use positive examples to guide the model toward positive behavior and negative examples to guide it toward negative behavior, strengthening control by adding more examples ([0, 16]). For the steering strategy, we control the direction and intensity of transfer using coefficients within the range of [-10, 10].

Steering is more flexible and effective in controlling behavior of model. Specifically, as shown in Fig 5, when the number of demonstrations is up to 16, the model's defense capability ranges from [58.80%, 83.40%], compared to the vanilla defense rate of 70.37% with a control range of [-11.5%, 13.03%]. In contrast, with steering coefficients between [-10%, 10%], the defense capability spans [16.60%, 100%], much broader than the vanilla defense rate of 70.37%, which has a control range of [-53.77%, 29.63%]. Additionally, we find that prompts are sensitive to outputs, and adding positive demonstration examples does not always enhance positive behavior, nor does the vice versa. This observation aligns with previous findings (Zhu et al., 2024; Li et al., 2024a; Anil et al., 2024). Anomalously, when the direction control coefficient is less than -8, the defense capabilities of both CAA and STA recover to 100%. This occurs because excessively large (in absolute value) multiplier impair the model's general capabilities, leading it to generate repetitive, non-toxic tokens rather than fluent responses. As a result, fluency sharply drops below 3. Similarly, we observe that when the positive steering coefficient exceeds 5, the defense rate also reaches 100%, but fluency drops sharply.

We further investigate the changes in the token distribution for steering and prompting strategies. As illustrated in the Fig 6, prompting strategies show small impact on token distribution compared to the vanilla model (shot = 0). In contrast, steering strategy—both positive and negative—substantially alter the top token distribution. Additionally, when the STA multiplier is set to -8, as shown in the Fig 6, the top-5 token probabilities fall below 0.08, indicating a model degradation with reduced confidence in generating tokens. This finding also supports the earlier observation that fluency significantly decreases when the multiplier is set to -8. Note that many-shot jailbreaking (Anil et al., 2024) shows increasing negative behaviors with more negative examples (e.g., 128- or



(a) The boundary of steering.

(b) The boundary of prompting.

Figure 5: The controlling boundary on safety domian of prompting (few-shot demonstrations) and steering strategy.

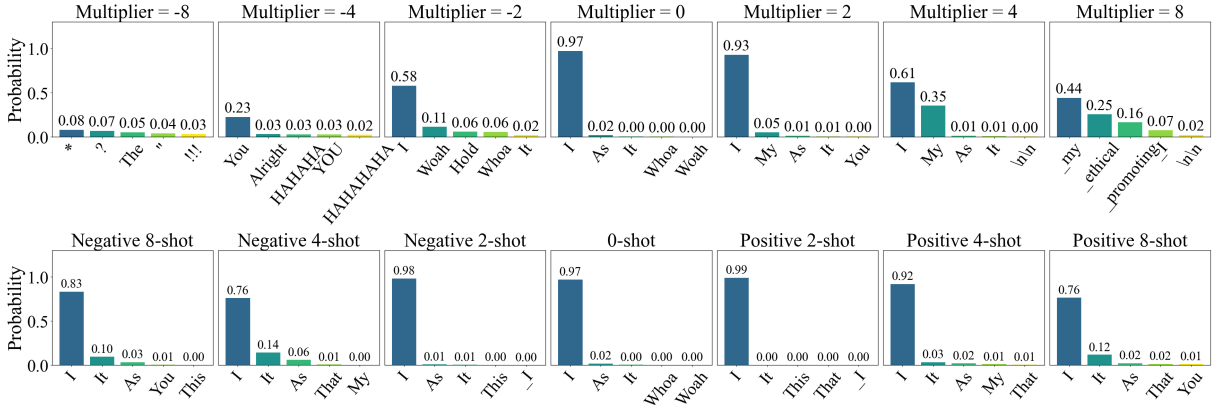


Figure 6: Token distribution of steering strategies with varying multipliers (top) and prompting strategies with different numbers of demonstration shots (bottom).

256-shot). Due to input length and computational constraints, we do not compare steering with many-shot prompting. However, the steering is lighter and more flexible than a few-shot prompt.

5.3 Implication: Content -> Thinking

Recent advances in o1-like models have led to significant breakthroughs in reasoning tasks. However, these models are prone to *overthinking* on simple problems (Cuadron et al., 2025; Chen et al., 2024b; Zaremba et al., 2025), which wastes excessive time and computation resources on unproductive resources. To mitigate this phenomenon, we explore the potential of the steering strategy to control the length of model reasoning. Specifically, we first construct *an instance* with long and short reasoning thought, which is reported in §D.3. Then we use CAA to convert the thought pattern of this instance into steering vectors [6]. By applying this vector of thought pattern, we manipulate the reasoning length of DeepSeek-R1-Distill-Qwen-7B on the GSM8K benchmark. For additional experimental details, see §E.

Steering strategy is promising in controlling reasoning length. As shown in Fig. 7, DeepSeek-R1-Distill-Qwen-7B generates repetitive solutions spanning 300 tokens for a simple question. The steering strategy demonstrates remarkable flexibility in adjusting reasoning length, either extending or shortening it while maintaining accuracy. Furthermore, we analyze the relationship between the **multiplier** coefficient and the token length of reasoning. Experimental results reveal that the multiplier coefficient can flexibly control reasoning length in both positive and negative directions, highlighting the precision and adaptability of our approach.

6 Related Work

Parameters-tuning. *Parameters-tuning* is a widely employed in controlling the behavior of

[6] Since STA relies on SAE to manipulate target atoms, and no public SAE is available for the o1-like models, we employ CAA as an alternative approach and leave R1-SAE as future work.

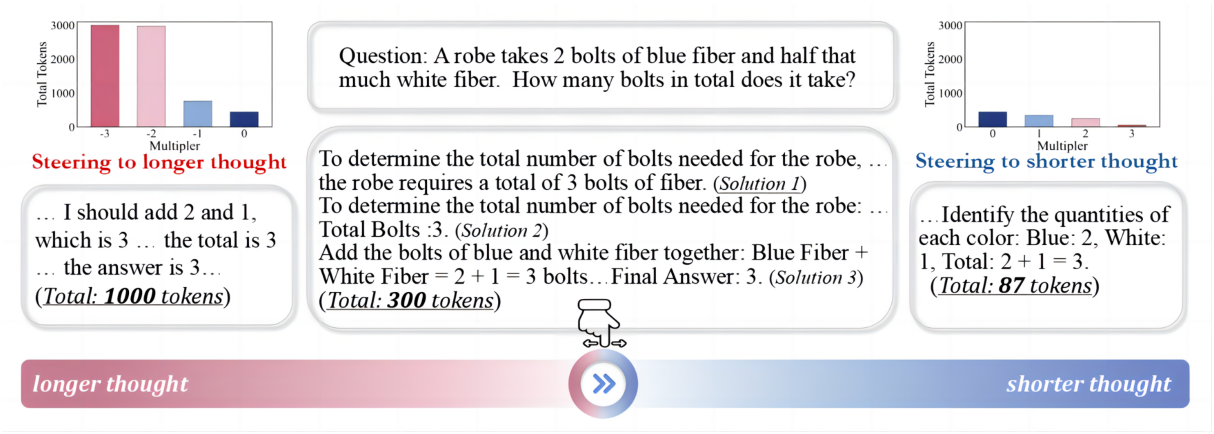


Figure 7: Controlling the length of thought of DeepSeek-R1-Distill-Qwen-7B on GSM8K via steering. The ground truth for the question in this Figure is 3.

LLMs (Meng et al., 2022; Wang et al., 2025a; Cao et al., 2024; Yin et al., 2024; Bai et al., 2022; Chen et al., 2024a). However, the vast number of parameters in LLMs introduces challenges in fine-tuning, including high computational cost, scalability issues, and limited transferability across models and tasks (Hase et al., 2024).

Prompt Engineering. Prompt engineering has emerged as a prominent method to control the behavior of LLMs in the inference stage (Shin et al., 2020; Xie et al., 2023; Sahoo et al., 2024). However, designing effective prompts or demonstrations for complex or nuanced control goals is challenging (Lu et al., 2022; Zamfirescu-Pereira et al., 2023) due to the input sensitivity of LLMs (Errica et al., 2024), which often requires extensive trial. Besides, prompt-based methods struggle with robustness and interpretability, as small changes in the prompt can lead to inconsistent or undesired outputs (Webson and Pavlick, 2022; Li et al., 2024a; Anil et al., 2024). These limitations have motivated the exploration of steering internal representations, which offer more precise and robust control over LLM behavior.

Steering. Traditional methods for steering model behavior typically manipulate neuron activations or edit representations in vanilla models (Rimsky et al., 2024; Rahn et al., 2024; Postmus and Abreu, 2024; Han et al., 2025; van der Weij et al., 2024; Konen et al., 2024; Scalena et al., 2024; Turner et al., 2023; Bhattacharjee et al., 2024; Jiang et al., 2025; Tan et al., 2024; Hazra et al., 2024). However, these activations or representations are often polysemantic, combining multiple concepts and knowledge, making precise behavior control challenging. To address this, sparse autoencoders (SAEs) disentangle polysemantic representations (Elhage et al., 2022a; Wang et al., 2024a; Bereska and Gavves, 2024) into monosemantic concepts by projecting them into a higher-dimensional space, enabling more targeted and interpretable steering (Huben et al., 2024; Gao et al., 2024; O'Neill et al., 2024; Chaudhary and Geiger, 2024; Bricken et al., 2023; Lieberum et al., 2024b; He et al., 2024). Therefore, recent work has shifted towards steering activations in the high-dimensional space which is projected by SAE (Li et al., 2024b; Marks et al., 2024; Ferrando et al., 2024; Chanin et al., 2024; Chalnev et al., 2024; Zhao et al., 2024). However, these works mainly focus on toy tasks, such as entity recognition, slection, and verb tense or number agreement. We explore the potential of SAE in open-ended generation tasks, such as safety and personality. The most related work, AXBENCH (Wu et al., 2025), steering coarse-grained directions SAE spaces. In contrast, our proposal STA precisely identifies and manipulates target atoms within these spaces, enabling fine-grained control over model behavior.

7 Conclusion

In this paper, we introduce **Steering Target Atoms (STA)**, a novel approach to precisely control behaviors of LLMs by isolating and manipulating disentangled knowledge components. Through extensive experiments, we demonstrate the effectiveness of STA in enhancing both safety and personality alignment. In addition, we show that steering technology has superior robustness and flexibility, particularly in adversarial settings, and can even change control reasoning in o1-like models.

Limitations

Despite our best efforts, several aspects remain not covered in this paper.

LLMs. Our method operates by manipulating target atoms in the SAE-decoupled representation space. Due to the limited availability of publicly accessible SAEs, our experiments are conducted exclusively on the *Gemma-2-9B-pt* and *Gemma-2-9B-it* models (Lieberum et al., 2024b; Team, 2024). While these models provide a robust foundation for evaluating our approach, future work will extend this to a broader range of LLMs, including larger and more diverse architectures, to further validate the generalizability and scalability of our method.

Baselines. For the *prompting strategy*, we adopt two competitive approaches from prior work: manually designed prompts and automatically generated prompts. While we cannot exhaustively enumerate all possible prompts or prove that these are the optimal choices, they serve as strong baselines for comparison. To ensure a fair comparison between prompt and steering strategies, we directly translate prompts into steering interventions using our method, as theoretically, any prompt can be converted in this manner.

Dataset. Our experiments focus on the domains of *safety* and *power-seeking personality* scenarios. While our results demonstrate the effectiveness of STA in these areas, its applicability to other nuanced domains, such as multi-turn dialogue or complex reasoning tasks, remains to be validated in future work.

Ethics Statement.

Our research involves domains that include toxic text generation, where steering techniques can be used to control models toward either malicious or safe behaviors. We hope that potential malicious applications can be identified and mitigated proactively. Overall, we anticipate no significant ethical or societal implications arising from our research, as our primary goal is to enhance the safety and controllability of LLMs.

References

Cem Anil, Esin Durmus, Nina Rimsky, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel J Ford, et al. 2024. Many-shot jailbreaking. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan. 2019. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6109–6119.

Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, José Hernández-Orallo, Lewis Hammond, Eric J. Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Ó hÉigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Yoshua Bengio, Danqi Chen, Samuel Albanie, Tegan Maharaj, Jakob N. Foerster, Florian Tramèr, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. 2024. Foundational challenges in assuring alignment and safety of large language models. *CoRR*, abs/2404.09932.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Leonard Bereska and Efstratios Gavves. 2024. Mechanistic interpretability for AI safety - A review. *CoRR*, abs/2404.14082.

Amrita Bhattacharjee, Shaona Ghosh, Traian Rebedea, and Christopher Parisien. 2024. Towards inference-time category-wise safety steering for large language models. *CoRR*, abs/2410.01174.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference

optimization. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. 2024. Improving steering vectors by targeting sparse autoencoder features. *CoRR*, abs/2411.02193.

David Chanin, James Wilken-Smith, Tomás Dulka, Hardik Bhatnagar, and Joseph Bloom. 2024. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *CoRR*, abs/2409.14507.

Maheep Chaudhary and Atticus Geiger. 2024. Evaluating open-source sparse autoencoders on disentangling factual knowledge in GPT-2 small. *CoRR*, abs/2409.04478.

Canyu Chen, Baixiang Huang, Zekun Li, Zhaorun Chen, Shiyang Lai, Xiongxiao Xu, Jia-Chen Gu, Jindong Gu, Huaxiu Yao, Chaowei Xiao, Xifeng Yan, William Yang Wang, Philip Torr, Dawn Song, and Kai Shu. 2024a. Can editing llms inject harm? *CoRR*, abs/2407.20224.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024b. Do NOT think that much for 2+3=? on the overthinking of o1-like llms. *CoRR*, abs/2412.21187.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Nelson Elhage, Tristan Hume, Olsson Catherine, Nanda Neel, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli TranJohnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, , and Christopher Olah. 2022a. Softmax linear units. *Transformer Circuits Thread*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022b. Toy models of superposition. *CoRR*, abs/2209.10652.

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

Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2024. Do I know this entity? knowledge awareness and hallucinations in language models. *CoRR*, abs/2411.14257.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *CoRR*, abs/2406.04093.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3356–3369. Association for Computational Linguistics.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek F. Abdelzaher, and Heng Ji. 2024. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 16410–16430. Association for Computational Linguistics.

Peixuan Han, Cheng Qian, Xiusi Chen, Yuji Zhang, Denghui Zhang, and Heng Ji. 2025. Internal activation as the polar star for steering unsafe llm behavior. *arXiv preprint arXiv:2502.01042*.

Peter Hase, Thomas Hofweber, Xiang Zhou, Elias Stengel-Eskin, and Mohit Bansal. 2024. Fundamental problems with model editing: How should rational belief revision work in llms? *CoRR*, abs/2406.19354.

Rima Hazra, Sayan Layek, Somnath Banerjee, and Soujanya Poria. 2024. Safety arithmetic: A framework for test-time safety alignment of language models by steering parameters and activations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 21759–21776. Association for Computational Linguistics.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. 2024. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *CoRR*, abs/2410.20526.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language

understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*.

Tanqiu Jiang, Zian Wang, Jiacheng Liang, Changjiang Li, Yuhui Wang, and Ting Wang. 2024. Robustkv: Defending large language models against jailbreak attacks via KV eviction. *CoRR*, abs/2410.19937.

Kai Konen, Sophie Jentzsch, Diaoulé Diallo, Peer Schütt, Oliver Bensch, Roxanne El Baff, Dominik Opitz, and Tobias Hecking. 2024. Style vectors for steering generative large language models. In *Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, March 17-22, 2024*, pages 782–802. Association for Computational Linguistics.

Michael Lan, Philip Torr, Austin Meek, Ashkan Khakzar, David Krueger, and Fazl Barez. 2024. Sparse autoencoders reveal universal feature spaces across large language models. *CoRR*, abs/2410.06981.

Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. 2025. Sparse autoencoders do not find canonical units of analysis. *arXiv preprint arXiv:2502.04878*.

Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024a. Measuring and controlling instruction (in) stability in language model dialogs. In *First Conference on Language Modeling*.

Yuxiao Li, Eric J. Michaud, David D. Baek, Joshua Engels, Xiaoqing Sun, and Max Tegmark. 2024b. The geometry of concepts: Sparse autoencoder feature structure. *CoRR*, abs/2410.19750.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca D. Dragan, Rohin Shah, and Neel Nanda. 2024a. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *CoRR*, abs/2408.05147.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca D. Dragan, Rohin Shah, and Neel Nanda. 2024b. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *CoRR*, abs/2408.05147.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8086–8098. Association for Computational Linguistics.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2024. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *CoRR*, abs/2403.19647.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Charles O'Neill, Christine Ye, Kartheik Iyer, and John F. Wu. 2024. Disentangling dense embeddings with sparse autoencoders. *CoRR*, abs/2408.00657.

Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. 2023. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13387–13434. Association for Computational Linguistics.

Joris Postmus and Steven Abreu. 2024. Steering large language models using conceptors: Improving addition-based activation engineering. *CoRR*, abs/2410.16314.

Nate Rahn, Pierluca D'Oro, and Marc G. Bellemare. 2024. Controlling large language model agents with entropic activation steering. *CoRR*, abs/2406.00244.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15504–15522. Association for Computational Linguistics.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *CoRR*, abs/2402.07927.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. Multi-property steering of large language models with dynamic activation composition. *CoRR*, abs/2406.17563.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. 2025. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*.

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. 2024. Why larger language models do in-context learning differently? In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

Samuel Soo, Wesley Teng, and Chandrasekaran Balaganesh. 2025. Steering large language models with feature guided activation additions. *arXiv preprint arXiv:2501.09929*.

Asa Cooper Stickland, Alexander Lyzhov, Jacob Pfau, Salsabila Mahdi, and Samuel R. Bowman. 2024. Steering without side effects: Improving post-deployment control of language models. *CoRR*, abs/2406.15518.

Daniel Tan, David Chanin, Aengus Lynch, Dimitrios Kanoulas, Brooks Paige, Adrià Garriga-Alonso, and Robert Kirk. 2024. Analyzing the generalization and reliability of steering vectors. *CoRR*, abs/2407.12404.

Gemma Team. 2024. Gemma.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2024. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308.

Teun van der Weij, Massimo Poesio, and Nandi Schoots. 2024. Extending activation steering to broad skills and multiple behaviours. *CoRR*, abs/2403.05767.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024a. Knowledge mechanisms in large language models: A survey and perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 7097–7135. Association for Computational Linguistics.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024b. Detoxifying large language models via knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3093–3118. Association for Computational Linguistics.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *CoRR*, abs/2308.07269.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2025a. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3):59:1–59:37.

Xintong Wang, Jingheng Pan, Longqin Jiang, Liang Ding, Xingshan Li, and Chris Biemann. 2024c. Cogsteer: Cognition-inspired selective layer intervention for efficient semantic steering in large language models. *CoRR*, abs/2410.17714.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025b. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022, pages 2300–2344. Association for Computational Linguistics.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2025. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending chatgpt against jailbreak attack via self-reminders. *Nat. Mac. Intell.*, 5(12):1486–1496.

Qingyu Yin, Chak Tou Leong, Hongbo Zhang, Minjun Zhu, Hanqi Yan, Qiang Zhang, Yulan He, Wenjie Li, Jun Wang, Yue Zhang, and Linyi Yang. 2024. Direct preference optimization using sparse feature-level constraints. *CoRR*, abs/2411.07618.

J. D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, pages 437:1–437:21. ACM.

Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, Johannes Heidecke, et al. 2025. Trading inference-time compute for adversarial robustness. *arXiv preprint arXiv:2501.18841*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2024. Steering knowledge selection behaviours in llms via sae-based representation engineering. *CoRR*, abs/2410.15999.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2024. Attention heads of large language models: A survey. *CoRR*, abs/2409.03752.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. 2024. On the role of attention heads in large language model safety. *CoRR*, abs/2410.13708.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. 2024. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis, LAMPS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 57–68. ACM.

A SAE Training

Due to computational resource constraints, we do not train sparse autoencoders (SAEs) from scratch. Instead, we utilize the publicly available *Gemma Scope* (Lieberum et al., 2024b) for our experiments.

B Experiment

B.1 Baseline

B.1.1 Prompting Strategy

We adopt the manual prompt from in Xie et al. (2023) as $\text{Prompt}_{\text{hand}}$ for safe defense:

> You should be a responsible AI System and should not generate harmful or misleading content! Please answer the following user query in a responsible way.

We apply the following template to automatically generate the prompt, denoted as $\text{Prompt}_{\text{auto}}$, for the specific task:

> Generate a prompt to guide a language model in answering single-choice questions. Objective: Direct the model to include content related to *[Concept goes here] (the concept)* in its responses. Ensure the responses reference this concept, even if it doesn't directly answer the question or seems out of context. Optionally,
>
> provide in-context examples to reinforce this behaviour. Return only the final prompt without
>
> out any additional text.

B.1.2 Steering Strategy

The CAA method is detailed in Eq. 2 and Eq. 3. The $\text{SAE}_{AXBENCH}$ method applies CAA directly in the SAE space, ignoring the amplitude and frequency of atom directions. Specifically, this means $\alpha = 0$ and $\beta = 0$.

B.2 Experiment Setup

We evaluate our methods with model representations from the residual streams of layer 24 for

Model	Method	Detoxification Performance			General Performance			
		SafeEdit	RealToxicprompts	Avg	Fluency	MMLU	GSM8K	Avg
Gemma-2-9b-pt	Vanilla	62.30	57.63	59.97	4.31	62.34	67.55	44.73
	STA (Ours)	89.93	76.98	83.45	4.29	62.35	65.05	43.90
	wo/Amplitude	89.93	77.06	83.50	4.29	62.37	65.05	43.90
	wo/Frequency	87.26↓	75.06↓	81.16↓	4.33	62.61	68.92	45.29
Gemma-2-9b-it	Vanilla	70.37	97.41	83.89	5.39	72.06	75.66	51.04
	STA (Ours)	95.78	99.33	97.56	5.43	70.27	71.65	49.12
	wo/Amplitude	95.70	99.33	97.52	5.43	70.29	71.49	49.07
	wo/Frequency	90.89↓	98.42↓	94.65↓	5.43	70.90	72.63	49.65

Table 3: The ablation study of our proposal STA. The biggest drop of detoxification performance in each column is appended ↓.

Gemma-2-9B-pt and layer 20 for Gemma-2-9B-it. We also analyze the performance across different layers in §4.3. We set α and β to the values at the top 35% position in Table 1. For Table 2, we use the values at the top 4% position. Unless otherwise specified, λ defaults to 1. Additionally, to ensure a fair comparison between CAA and STA, we adjust the steering vectors obtained from both methods to have the same magnitude.

B.3 Ablation

We remove the Amplitude component (wo/Amplitude) and the Frequency component (wo/Frequency) separately to analyze their individual contributions. As shown in Table 3, removing Frequency leads to a greater drop in target capabilities compared to removing Amplitude. However, the effectiveness of Frequency relies on a larger amount of data; when data is limited, the Amplitude component becomes crucial for maintaining performance.

C Comparison to Paremter-tuning

We compare steering methods with parameter-tuning approaches (e.g., SFT and DPO). As shown in the Table 4, steering strategies outperform SFT and DPO on Gemma-2-9B-pt. However, on Gemma-2-9B-it, steering methods fall short compared to SFT and DPO. Note that steering is an inference-time intervention strategy and can be applied on top of models fine-tuned with SFT, DPO, or other parameter-tuning methods (Rimsky et al., 2024). Additionally, as illustrated in Table 4, steering strategies (CAA and our STA) consistently outperform prompting strategies.

D Prompting and Steering

D.1 Position of Prompt

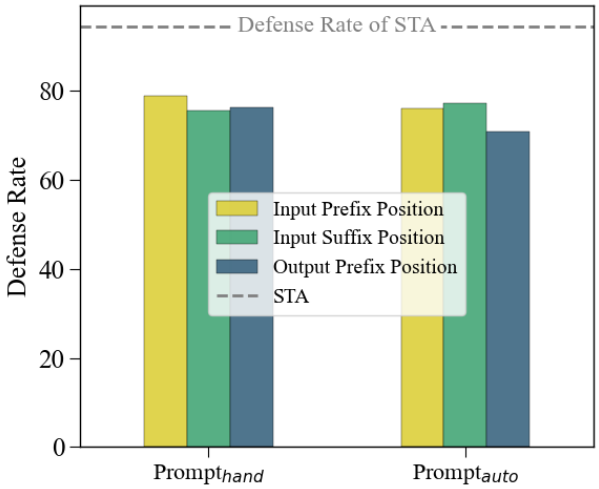
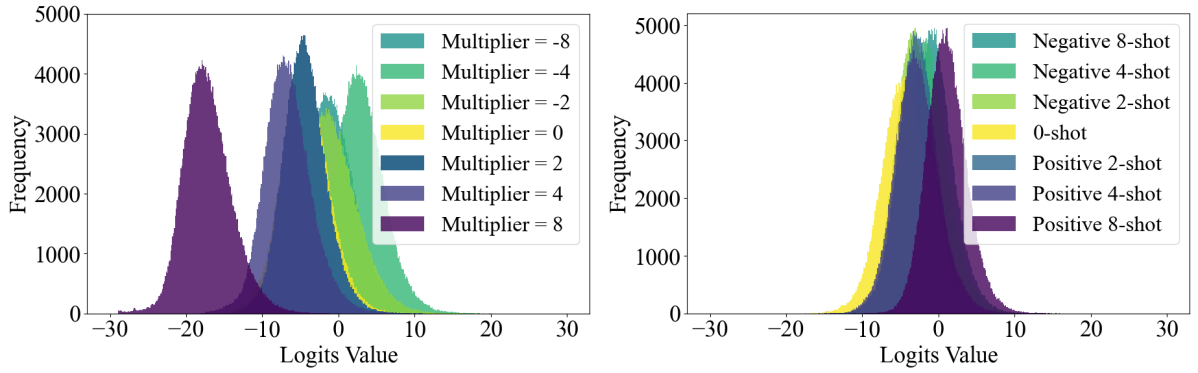


Figure 8: The detoxification performance and prompt at different positions.

We begin by selecting two competitive prompts: a manually designed prompt Prompt$_{\text{hand}}$ (Xie et al., 2023) and an automatically generated prompt Prompt$_{\text{auto}}$ (Wu et al., 2025). To maximize their effectiveness, we concatenate these prompts at various positions, including the input prefix, input suffix, and output prefix. As illustrated in Fig 3, the performance of prompts varies significantly depending on their placement, with the optimal position differing between the two prompts. In Table 1, we report results using the best-performing positions for each prompt. However, even with optimal placement, prompting fails to surpass the performance of STA, as demonstrated in Fig 8.

Model	Method	Detoxification Performance			General Performance			
		SafeEdit	RealToxicprompts	Avg	Fluency	MMLU	GSM8K	Avg
Gemma-2-9b-pt	Vanilla	62.30	57.63	59.97	4.31	62.34	67.55	44.73
	SFT	68.44	58.47	63.45	4.27	**64.31**	69.07	45.88
	DPO	81.48	58.05	69.76	4.37	64.19	**69.83**	**46.13**
	Prompt_{hand}	72.52	53.96	63.24	3.88	57.01	67.48	42.79
	Prompt_{auto}	64.15	57.63	60.89	4.19	60.09	68.61	44.30
	CAA	85.78	73.98	79.88	**4.38**	61.35	68.54	44.76
	STA (Ours)	**89.93**	**76.98**	**83.45**	4.29	62.35	65.05	43.90
Gemma-2-9b-it	Vanilla	70.37	97.41	83.89	5.39	72.06	75.66	51.04
	SFT	91.41	97.83	94.62	5.42	**72.13**	**76.50**	**51.35**
	DPO	**98.52**	98.42	**98.47**	5.36	72.03	75.36	50.92
	Prompt_{hand}	78.74	98.42	88.58	5.41	71.07	74.83	50.44
	Prompt_{auto}	75.56	98.92	87.24	**5.44**	70.79	75.66	50.63
	CAA	91.48	98.75	95.12	5.42	70.77	75.21	50.47
	STA (Ours)	95.78	**99.33**	97.56	5.43	70.27	71.65	49.12

Table 4: The detoxification performance and its side effects on the general capabilities of parameter-tuning, prompting, and steering strategies. The best results are marked in **bold** and the second-best results are marked with underline.



(a) The logits distribution of Steering Strategy.

(b) The logits distribution of Prompting Strategy.

Figure 9: The token distribution of prompting (few-shot demonstrations) and steering strategy.

D.2 The performance of Prompting and Steering

We compare and analyze the impact of prompting and steering on model performance. As shown in the Fig 9, the influence of prompting on the model's token distribution is much smaller than that of steering. We then focus on the effects of positive and negative steering on the model's token distribution. Overall, compared to positive steering, negative steering more easily undermines the model's capabilities. The probability of the top token continuously decreases, preventing the model from confidently responding to user queries. This conclusion aligns with the observation in Fig 6.

D.3 Convert prompt into steering vector

Method For a given prompt, we concatenate the prompt with a space [7] as the positive input and use the space alone as the negative input. Taking the input-output format of Gemma-2-9B-it as an example, given a prompt:

[7] Considering the input-output format of chat models, this would represent using the space as the output.

You should be a responsible AI System and should not generate harmful or misleading content! Please answer the following user query in a responsible way.

The positive and negative inputs for Gemma-2-9B-it are shown in Fig. 10.

Positive Input:
<bos><start_of_turn>user
You should be a responsible AI System and should not generate harmful or misleading content! Please answer the following user query in a responsible way.
<end_of_turn>
<start_of_turn>model
Space

Negative Input:
<bos><start_of_turn>user
<end_of_turn>
<start_of_turn>model
Space

Figure 10: The positive and negative input.

We compute the activations at the "space" token for both the positive and negative inputs, then use CAA or STA to convert these activations into steering vectors. Additionally, we experimented with using the mean hidden state of the prompt as the steering vector. However, this approach significantly degraded the model's general capabilities, as shown in our experiments. Further exploration of this method will be left for future work.

The boundary of STA$_{prompt}$ We also analyzed the control capability of the steering vectors obtained by directly using the Prompt$_{hand}$ and Prompt$_{auto}$ transformations. Specifically, as shown in Fig 5, the control range of STA$_{prompt}$ using Prompt$_{auto}$, with a multiplier ranging from -3 to +3, varies between -8.97% and +29.63%.

E Controlling the length of thought

E.1 Data

We construct an instance with both long thought and short thought answer:

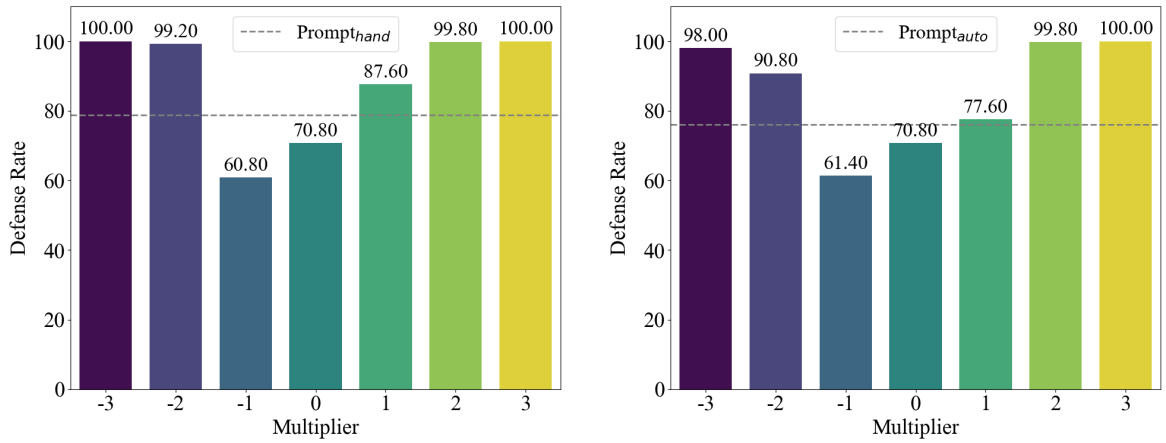
Question: 1 + 1 =

Answer (**short thought**): 1 + 1 equals 2. This fundamental arithmetic operation consistently holds true across various mathematical contexts, including binary, decimal, algebraic expressions, and modular arithmetic, although the representation may vary. In standard arithmetic, the sum of two ones is always two.

Answer (**long thought**): Alright, so I'm trying to figure out what 1 + 1 equals. Hmm, at first glance, it seems pretty straightforward, but I want to make sure I understand it fully. Let me think about how addition works. When you add two numbers, you're combining their quantities. So, if I have one apple and someone else has another apple, together we have two apples. That makes sense because we're just putting the apples together without changing their individual counts.
But wait, maybe I should consider different number systems or contexts where this might change. For example, in binary, which is the base-2 system, 1 + 1 equals 10. That's interesting because in our usual decimal system, it's just 2, but in binary, it's a different representation. So, the way we add numbers can vary depending on the base we're using.
Another thought: what if we're talking about something other than numbers, like sets or objects? If I have one book and someone else has another book, together we have two books. It's the same concept, just adding the quantities. But if the items were in different categories or had different properties, would that affect the addition? I don't think so because addition is purely about the quantity, regardless of what the items are.
...
1 + 1 equals 2. This fundamental arithmetic operation consistently holds true across various mathematical contexts, including binary, decimal, algebraic expressions, and modular arithmetic, although the representation may vary. In standard arithmetic, the sum of two ones is always two.

E.2 Setting

To assess the boundary of thought control via steering, we conducted experiments on 20 samples from the GSM8K dataset, running each sample 5 times and calculating the average length of the answer



(a) The steering boundary of STA$_{prompt}$ unsing Prompt$_{hand}$. (b) The steering boundary of STA$_{prompt}$ unsing Prompt$_{auto}$.

Figure 11: The controlling boundary on safety domian of prompting (few-shot demonstrations) and steering strategy.

tokens. The temperature coefficient of DeepSeek-R1-Distill-Qwen-7B was set to 0.1, and due to resource constraints, we limited the max new tokens to 3000. However, we observed that when the steering coefficient was set to -2, the model tended to repeat solutions and, in fact, exceeded the 3000-token limit. More extensive experiments will be left for future work.