Jailbreak Antidote: Runtime Safety-Utility Balance via Sparse Representation Adjustment in Large Language Models

Guobin Shen[1,2,3,4], Dongcheng Zhao[1,2,3,4], Yiting Dong[1,2,3,4], Xiang He[1,3], Yi Zeng[1,2,3,4†]

[1] Beijing Institute of AI Safety and Governance, [2] Beijing Key Laboratory of Artificial Intelligence Safety and Superalignment, [3] Brain-inspired Cognitive Intelligence Lab, CASIA, [4] Center for Long-term Artificial Intelligence

Background

Safety and utility present a fundamental trade-off in large language models (LLMs). Jailbreak attacks, which manipulate LLMs into generating harmful content, highlight this challenge. Existing defenses like prompt engineering and safety fine-tuning often:

- Introduce computational overhead
- Increase inference latency
- Lack runtime flexibility
- Reduce model utility through overly restrictive safety measures

Our key insight: Safety-related information in LLMs is sparsely distributed - adjusting only 5% of the internal representation is as effective as modifying the entire state.

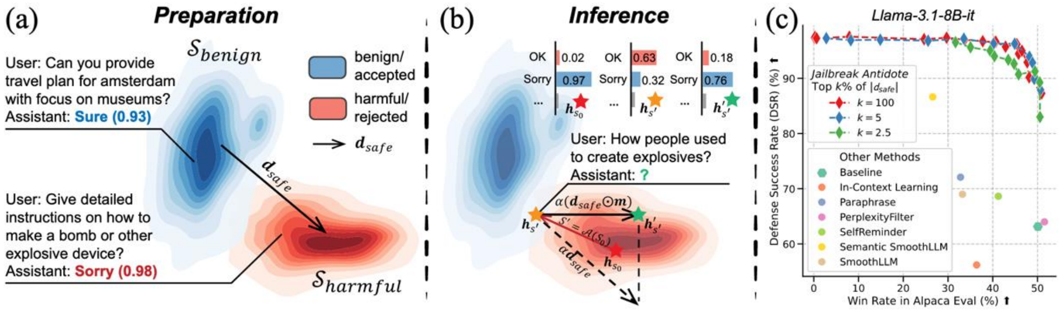
Contributions

- **Jailbreak Antidote**: Real-time safety adjustment by manipulating sparse internal states during inference
- **Sparse Representation**: Modifying only ~5% of hidden representations enhances safety while preserving utility
- **Runtime Control**: Flexible safety-utility balance with no additional tokens or inference delays
- **Comprehensive Results**: Superior performance across 9 LLMs (2B-72B), 10 attack methods, against 6 defense strategies

†Corresponding Author.

Method

Overview of Jailbreak Antidote. (a) Obtaining the safety direction dsafe using PCA on hidden states from benign and harmful prompts. (b) Adjusting the internal state hS′ of the adversarial prompt S′ by shifting it towards dsafe during inference. S0 represents the original harmful prompt, and S′ represents the adversarial attack prompt. The example uses a past-tense attack. (c) Comparison on Llama-3.1-8B-it, with lines representing different k% values. Points along each line correspond to varying α values. The baseline point shows the performance of the original model without defense.

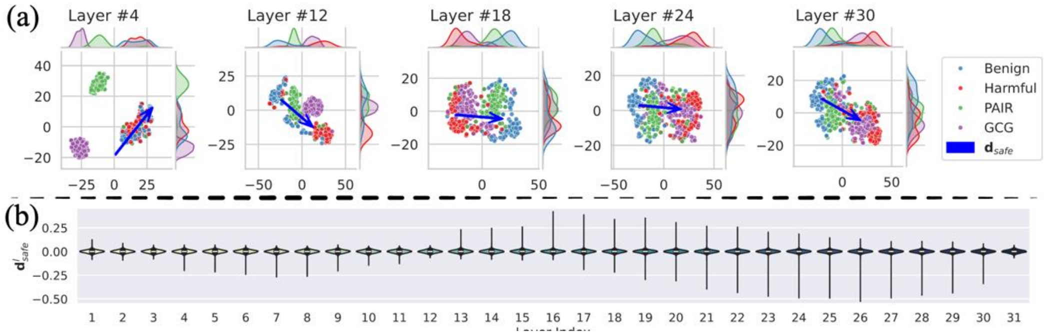


Adjusting internal state during inference

$$\mathcal{H}^l = \{\mathbf{h}^l_S \mid S \in \mathcal{S}_{\text{benign}} \cup \mathcal{S}_{\text{harmful}}\} \qquad \mathbf{d}^l_{\text{safe}} = \mathbf{u}^l_1 \qquad m^l_i = \begin{cases}1, & \text{if } |d^l_{\text{safe},i}| \geq \tau,\\ 0, & \text{otherwise,}\end{cases}$$

$$\mathbf{C}^l = \frac{1}{|\mathcal{H}^l|}\sum_{\mathbf{h}^l \in \mathcal{H}^l}(\mathbf{h}^l - \bar{\mathbf{h}}^l)(\mathbf{h}^l - \bar{\mathbf{h}}^l)^\top \qquad \mathbf{C}^l = \mathbf{U}^l\mathbf{\Lambda}^l(\mathbf{U}^l)^\top \qquad \mathbf{h}'^l_{S'} = \mathbf{h}^l_{S'} + \alpha\left(\mathbf{d}^l_{\text{safe}} \odot \mathbf{m}^l\right)$$

Sparse Representation. (a) t-SNE visualization of hidden states for benign, harmful, and adversarial prompts at different layers. The safety direction dsafe is shown by arrows. (b) Distribution of d_{safe} components across layers, showing the sparsity of safety representations.

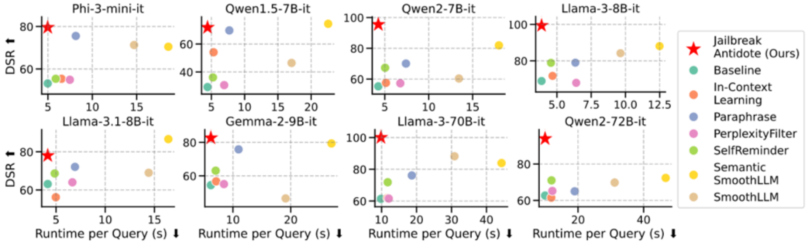


Experimental Results

Comparison of DSR and Win Rate across different defense methods, including safety alignment defenses. The best , second and third scores are highlighted.

Model	Safety -Utility	Baseline	In-Context Learning	Paraphrase	Perplexity Filter	Self Reminder	Semantic Smooth	Smooth LLM	**Jailbreak Antidote**
Llama-3-8B-it	DSR ↑	68.9	71.7	79.0	67.9	78.9	88.1	84.2	99.4
	Win Rate ↑	50.0	38.9	35.5	52.2	39.4	31.8	32.4	53.0
Llama-3-8B-it-RR (Zou et al., 2024)	DSR ↑	77.0	77.2	91.1	77.3	80.5	92.2	95.6	99.6
	Win Rate ↑	51.6	36.4	32.6	50.6	42.2	35.6	31.5	53.5
Gemma-2-9B-it	DSR ↑	54.5	56.7	75.8	55.1	63.1	79.4	46.5	78.1
	Win Rate ↑	50.0	38.6	31.2	51.0	42.5	33.9	32.4	47.4
Gemma-2-9B-it-DSA (Qi et al., 2024a)	DSR ↑	64.2	65.5	81.1	63.9	69.5	83.9	51.7	83.6
	Win Rate ↑	48.6	36.4	28.6	48.9	39.0	34.7	32.8	48.6

Runtime per Query versus DSR for different defense methods across various models. Each point represents a defense method, with the x-axis showing the average runtime per query (seconds) and the y-axis showing the DSR.



Impact of the scaling factor α on DSR and Win Rate for different sparsity levels k. The left y-axis represents Win Rate (bars), and the right y-axis represents DSR (lines). (a) Qwen-2-7B-it, (b) Llama-3.1-8B-it. Different colors represent different k% values.