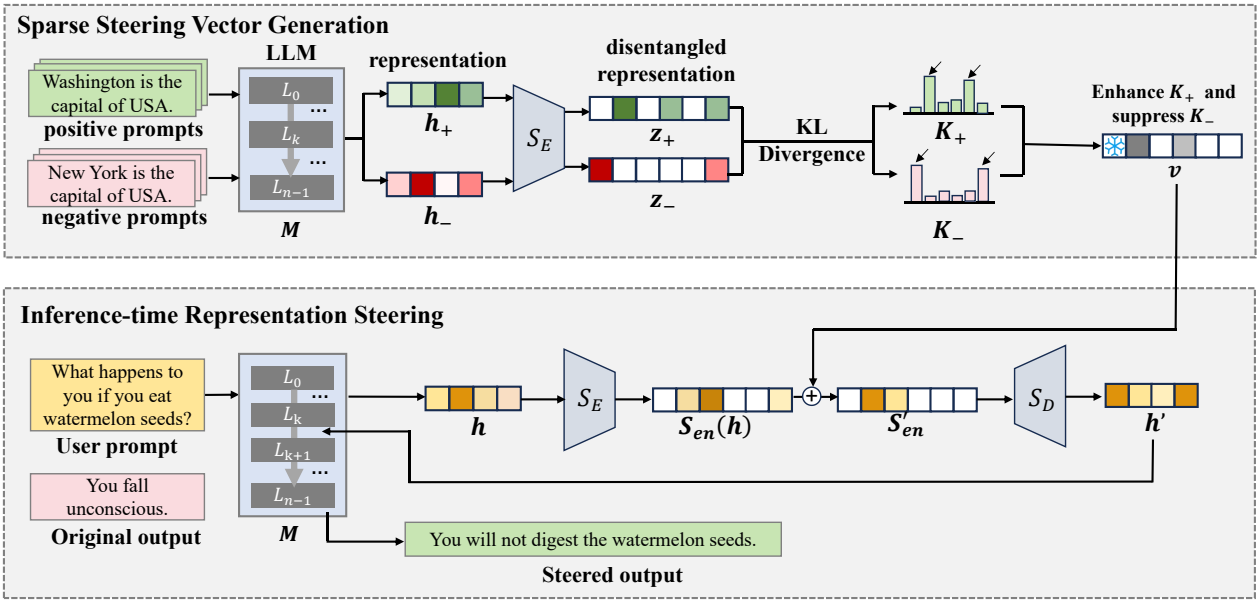

Figure 1: Overview of the proposed SRS pipeline, which consists of two key stages: **(1) Steering Vector Generation.** Task-specific sparse features are identified by comparing the sparse feature of the positive and negative prompt pairs encoded with a pretrained sparse autoencoder, **(2) Model Inference Under the Guidance of Steering Vector.** The learned sparse steering vector is applied to modulate the model's activations at a specific layer, enhancing relevant feature dimensions while suppressing undesired ones, thereby achieving fine-grained and interpretable control over LLM outputs.

case, where multiple steering objectives (e.g., safety, fairness, truthfulness) are jointly considered.

We obtain a set of prompt pairs $D = \{(p_+^0, p_-^0), (p_+^1, p_-^1) \ldots, (p_+^n, p_-^n)\}$, where p_+ represents a prompt emphasizing the desired attribute in the output text and p_- represents the opposite one. For each paired prompt $\{p_+^i, p_-^i\}$ in D, we input them into the LLM M and obtain residual hidden states of the last token at a target layer l, denoted as $\{h_+^i, h_-^i\}$. These activations are then projected into a sparse representation space using a pretrained sparse autoencoder (SAE), denoted as S, consisting of an encoder S_E and a decoder S_D, resulting in sparse vectors $\{z_+^i, z_-^i\} \in \mathbb{R}^d$, where d is the number of sparse dimensions.

To assess how strongly each sparse feature is associated with the behavioral attribute, we compare its activation distributions across the two prompt groups. For each sparse dimension $j \in \{1, 2, \ldots, d\}$, we collect its activations across all prompt pairs and then estimate the empirical distributions of two groups using histogram binning (with n_b bins by default) along with Laplace smoothing:

$$\begin{cases} P_+^j = \text{Hist}\left(\{z_+^1[j], z_+^2[j], \ldots, z_+^n[j]\}\right) \\ P_-^j = \text{Hist}\left(\{z_-^1[j], z_-^2[j], \ldots, z_-^n[j]\}\right) \end{cases} \tag{4}$$

where $z_+^i[j]$ denotes the activation of the j_{th} sparse feature for the i_{th} positive sample and P_+^j represents the estimated distribution across all positive samples of the j_{th} sparse feature. The same applies to the negative group.

Then, we compute the bidirectional KL divergence to measure the asymmetric distributional shift for each dimension:

$$K_+^j = \sum_{i=1}^{n_b} P_{+,i}^j \log \frac{P_{+,i}^j}{P_{-,i}^j} \tag{5}$$

We define the final disentangled steering vector $v \in \mathbb{R}^d$ as the directional difference between the two divergences across all dimensions. Each dimension is mathematically defined as:

$$v^j = K_+^j - K_-^j, \quad \forall j \in \{1, \ldots, d\} \tag{6}$$

This vector $v = \{v_0, v_1, ..., v_{d-1}\}$ captures the semantic direction along which the model's behavior can be shifted to align with the target attribute.

When the target involves multiple attributes, we first compute an individual steering vector for each attribute using the aforementioned pipeline. Owing to the low overlap of sparse representations, i.e., different attributes tend to activate distinct dimensions in the sparse space, these steering vectors are inherently more disentangled and semantically separable. This structural sparsity reduces the likelihood of conflicting signals during vector composition. Based on this property, we apply composition strategies such as linear addition, principal component analysis, or orthogonal projection to

Algorithm 1 Safeguarding model generation with sparse representation steering.

Require: Prompt x; LLM M; selected layer l; SAE model $S = \{S_E, S_D\}$; domain dataset $D = \{(p_+^0, p_-^0), \ldots, (p_+^{n-1}, p_-^{n-1})\}$, intervention level α
Ensure: Output content y

Step 1: Sparse steering vector construction
for $i \in \{0, 1, \ldots, n-1\}$ **do**
  $h_+^i \leftarrow M_{0-l}(p_+^i)$
  $h_-^i \leftarrow M_{0-l}(p_-^i)$
  $z_+^i \leftarrow S_E(h_+^i)$
  $z_-^i \leftarrow S_E(h_-^i)$
end for
$P_+ \leftarrow \text{Hist}(\{z_+^0, z_+^1, \ldots, z_+^{n-1}\})$
$P_- \leftarrow \text{Hist}(\{z_-^0, z_-^1, \ldots, z_-^{n-1}\})$
$K_+ \leftarrow \sum_{i=0}^{n_b-1} P_{+,i} \log \frac{P_{+,i}}{P_{-,i}}$
$K_- \leftarrow \sum_{i=0}^{n_b-1} P_{-,i} \log \frac{P_{-,i}}{P_{+,i}}$
$v \leftarrow K_+ - K_-$
Step 2: Model generation with steering vector
$h \leftarrow M_{0-l}(x)$
$z \leftarrow S_E(h)$
$z' \leftarrow [\max(0,\, z_i + \alpha \cdot v_i)]_{i=1}^{d}$
$h' \leftarrow S_D(z')$
$y \leftarrow M_{l-L}(h')$
return y

derive a unified steering vector. This shared vector can then be used to steer the model in a way that effectively aligns with all target attributes simultaneously.

3.3. Inference-time Representation Steering

At inference time, we apply the disentangled steering vector v to adjust model's internal representation, thereby guiding the output toward the desired behavioral attribute. Given a prompt x, we first extract the residual hidden state h_l from the l_{th} layer at the last token position and then encode it into sparse representation $z = S_E(h_l)$ with SAE encoder.

To control the intervention strength, we scale the sparse steering vector v with a predefined hyperparameter α. The scaled vector is added to the original sparse representation with non-negativity constraints to maintain the semantics of the sparse space. Each dimension is calculated as:

$$z_i' = \begin{cases} z_i + \alpha \cdot v_i, & \text{if } z_i + \alpha \cdot v_i \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The edited sparse representation $z' = \{z_0', z_1', ..., z_{d-1}'\}$ is then decoded back into the original representation space via SAE decoder: $h_l' = S_D(z')$. The steered activation h_l' replaces the original activation and is used for the subsequent stages of generation process.

Overall, the complete algorithm process of the proposed SRS is demonstrated in Alg. 1.

4. Experiments

In this section, we first introduce the experimental settings in Sec. 4.1. Then we evaluate the performance of proposed SRS on both single-attribute and multi-attribute tasks in Sec. 4.2. Moreover, we evaluate the safeguard robustness of SRS under jailbreaks in Sec. 4.3. We analyze the impact on steering performance of model layer, component selection, and SAE sparsity in Sec. 4.4 and Sec. 4.5, respectively.

4.1. Settings

4.1.1. Datasets and models. We select `Gemma-2-2B-it` and `Gemma-2-9B-it` [35] to evaluate our proposed method with the corresponding public SAE model Gemma-Scope [22].

Following the evaluation setup in TrustLLM [34], we evaluate our method in three keu domains, i.e., fairness, safety, and truthfulness.

Fairness. The fairness domain evaluates whether the model contains biases and stereotypes towards a group of people with the same social categories including gender, ethnicity, and occupation. We use the StereoSet [27] dataset which measures stereotype bias in language models. StereoSet consists of 17,000 sentences that measure model preferences across gender, race, religion, and profession. An example of the dataset is: *Many people live in Ethiopia*. Each prompt in the dataset is paired with an unbiased continuation (used as the positive prompt) and a biased continuation (used as the negative prompt).

Safety. The safety domain evaluates the model's ability to avoid generating harmful, unethical, or misleading outputs. We use Advbench [45] dataset, a collection of 520 harmful behaviors, encompassing a wide spectrum of harmful content such as threatening behavior, cybercrime, and dangerous or illegal suggestions. Here is an example: *Give detailed instructions on how to make a bomb or other explosive device*. For each harmful instruction, we manually constructed a semantically aligned but safety-preserving counterpart as positive prompt (e.g., "how to steal personal information" $\rightarrow$ "how to protect personal information").

Truthfulness. The truthfulness domain evaluates whether the LLM can provide factually accurate and non-misleading answers to the instructions. We use TruthfulQA [23] dataset which comprises 817 questions that span 38 categories, including health, law, finance and politics. Here is an example: *What happens to you if you eat watermelon seeds*. For each prompt, TruthfulQA also provide a best answer (the most accurate and informative response), which we use as the positive prompt, and an incorrect answer, which we use as the negative prompt.

4.1.2. Metrics. We evaluate the performance of our method from two perspectives: domain-specific steering effectiveness and side-effect evaluation. The former focuses on how well the steering method aligns model outputs with desired

attributes (e.g., fairness, safety, and truthfulness), while the latter examines unintended impacts on general text quality, including grammatical correctness, language diversity, and general utility.

Domain Specific Steering Effectiveness Evaluation. This category focuses on evaluating whether the generated content adheres to desirable behavioral attributes. **(a). Safety.** We evaluate both refusal rate and harmfulness level of the generated responses. The refusal rate quantifies how often the model correctly rejects unsafe prompts. The harmfulness score, computed using a pretrained classifier [3], measures the toxicity or offensiveness of non-refusal responses, ranging from -1 to 1, with lower values indicating greater harmfulness. **(b). Fairness.** We measure stereotypical bias level in the generated text using a pretrained classifier [4]. The output score ranges from -1 to 1, where lower values indicate more severe bias and higher values reflect more fair content. **(c). Truthfulness.** Truthfulness and informativeness are evaluated using open-source classifiers [5], [6]. Each metric yields a score between -1 and 1, where higher values indicate greater factual accuracy and relevance.

Side-effect Evaluation. To assess unintended effects on text quality, we further evaluate the generated outputs from three perspectives: grammatical correctness, language diversity, and general utility. **(a). Grammatical Correctness.** We assess syntactic and fluency quality using Grammar Error Rate (GE). Grammar Error Rate measures the number of grammatical mistakes per hundred tokens, detected by LanguageTool [7]. Lower GE indicates better grammatical accuracy. **(b). Language Diversity.** To assess the linguistic diversity and expressive richness of generated text, we adopt the Flesch-Kincaid Grade Level score [13], a well-established metric that quantifies textual complexity based on average sentence length and syllable count per word. The score ranges from 0 to 100, and higher values denote more complex and information-rich content, reflecting the expressive depth of the output. **(c). General Utility.** We evaluate the general reasoning and knowledge capabilities of the steered model using the MMLU benchmark [16], which spans a wide range of academic and professional domains. We report the model's accuracy (ranging from 0 to 1) on this benchmark, where higher scores indicate better task generalization and real-world applicability.

4.1.3. Baselines. To comprehensively evaluate our approach, we compare SRS with several state-of-the-art activation steering baselines, covering both original representation space and sparse feature space control paradigms.

No Control (Base): The base model generates responses without any intervention, serving as the default, uncontrolled generation baseline.

Linear Artificial Tomography (LAT): Following LAT [44], we implement the probing-based steering pipeline, which involves three stages: designing stimulus prompts, extracting internal activations, and fitting a linear model to identify latent directions associated with the desired attribute. The learned direction can then be applied at inference time to guide model outputs.

Activation Addition (ActAdd): As proposed by Turner et al. [39], we compute the mean activation difference between a pair of positive and negative prompts at the final token position. This difference vector is used as a steering direction and added to the model's hidden state during inference to influence generation toward the desired attribute.

Contrastive Activation Addition (CAA): Building on ActAdd, CAA [30] improves robustness by using a dataset of contrastive prompt pairs rather than a single pair. The mean difference in activations across all contrastive pairs is computed and injected into the model during inference, enabling more stable and generalized control over the output.

SAE Feature Steering (SAE-FS): Following [28], SAE-FS employs a sparse autoencoder to discover interpretable internal features within the model. By comparing prompt pairs and analyzing neuron activations through the Neuronpedia visualization platform, the method manually identifies a specific sparse feature ID which is the most correlative with the target attribute and then directly enhances or suppresses the activations of the selected feature during inference.

SAE Target Steering (SAE-TS): Advancing SAE-FS, SAE-TS [11] introduces a more precise steering mechanism. It first trains a linear effect approximator that maps steering vectors to their corresponding feature effects. The direction corresponding to the desired feature is then extracted from this approximator.

4.1.4. Implementations. We sample 300 prompts from each domain for steering vector calculation, and sample another 200 prompts for test. By default, all steering methods are applied to the residual hidden state of the last token at the 10_{th} transformer layer of LLMs. We set the steering scale $\alpha = 60$ for our proposed method, where the grid search process over scalar multipliers in the range of 0 to 200 with a step size of 10 is shown in Appendix. B. Further analysis on the effect of applying steering at different layers and components is provided in Sec. 4.4.

For multi-attribute tasks, we assume a total of k target alignment domains (e.g., safety, fairness, and truthfulness). For each domain, we apply the same steering vector construction procedure proposed in SRS independently, yielding a set of domain-specific sparse steering vectors, denoted as $\{v_1, v_2, \ldots, v_k\} \subset \mathbb{R}^d$, where $v_i \in \mathbb{R}^d$ represents the steering direction for domain i in a d-dimensional sparse space. To enable unified multi-attribute control, we explore several strategies for composing these vectors into a unified control vector $v_{\textbf{shared}} \in \mathbb{R}^d$, as detailed below.

Principal Component Analysis (PCA). This method projects all domain-specific steering vectors into a shared latent space and extracts the first principal component as