Signal in the Noise: Polysemantic Interference Transfers and Predicts Cross-Model Influence

Bofan Gong[*]
Independent Scholar

Shiyang Lai[*]
University of Chicago

Dawn Song
UC Berkeley

Abstract

Polysemanticity is pervasive in language models and remains a major challenge for interpretation and model behavioral control. Leveraging sparse autoencoders (SAEs), we map the polysemantic topology of two small models (`Pythia-70M` and `GPT-2-Small`) to identify SAE feature pairs that are semantically *unrelated* yet exhibit interference within models. We intervene at four loci (prompt, token, feature, neuron) and measure induced shifts in the next-token prediction distribution, uncovering polysemantic structures that expose a systematic vulnerability in these models. Critically, interventions distilled from *counterintuitive* interference patterns shared by two small models transfer reliably to larger instruction-tuned models (`Llama-3.1-8B/70B-Instruct` and `Gemma-2-9B-Instruct`), yielding predictable behavioral shifts without access to model internals. These findings challenge the view that polysemanticity is purely stochastic, demonstrating instead that interference structures generalize across scale and family. Such generalization suggests a convergent, higher-order organization of internal representations, which is only weakly aligned with human intuition and structured by latent regularities, offering new possibilities for both black-box control and theoretical insight into human and artificial cognition. Code and data are available here.

1 Introduction

Polysemanticity refers to the phenomenon in which individual neurons or groups of neurons in neural networks often encode a greater number of distinct features or concepts than the number of neurons involved. This property becomes increasingly prevalent as models scale and has been shown to enhance learning performance (Wang et al., 2024; Marshall & Kirchner, 2024; Oikarinen & Weng, 2024b). Anthropic's work on *superposition* builds on prior insights, showing that large transformer models encode more features than neurons by using linear combinations of activations. This mechanism sacrifices monosemanticity but significantly improves model capability (Elhage et al., 2022). Mathematical analyses reveal that polysemantic neurons enable networks to represent exponentially more features compared to monosemantic approaches (Elhage et al., 2022).

However, this representational efficiency comes with notable trade-offs. Most significantly, it complicates model interpretability, as entangled representations obscure how human-understandable concepts are encoded within the model's internal structure. One mechanistic approach to address this challenge is the use of sparse autoencoders (SAEs), which aim to disentangle superimposed features by learning sparse, higher-dimensional representations of model activations. SAEs enable the extraction of interpretable, monosemantic features, where each SAE neuron ideally corresponds to a single concept (Bricken et al., 2023; Templeton et al., 2024)[2]. Recent work has shown that SAE-derived features exhibit a degree of universality across different LLMs (Lan et al., 2024), suggesting

[*] Equal contribution, alphabetical ordered. Correspondence to `shiyanglai@uchicago.edu` and `bfangong@gmail.com`.

[2] Nevertheless, several studies have also documented limitations of SAEs (see Appendix N).

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

the existence of fundamental patterns in how neural networks encode meaning. This consistency hints at the emergence of shared semantic topologies that persist across architectures and training regimes, raising profound questions about whether these patterns are merely computational artifacts or reflections of latent semantic regularities (Huh et al., 2024). Except for SAEs, a broader range of interpretability techniques is emerging at the same time (Chang et al., 2025; Dunefsky et al., 2024).

The second trade-off, which is largely overlooked in current literature, involves systematic vulnerability stemming from polysemantic structures in language models. In Anthropic's toy experiments, they note that stronger superposition can make models more vulnerable to adversarial attacks (Elhage et al., 2022). Beyond this, to our knowledge, there is very little existing empirical research that directly addresses the safety implications of polysemanticity in language models. In contrast, the vision model domain has a well-established body of work on various forms of adversarial model control that exploit polysemantic representations (Goh et al., 2021; Oikarinen & Weng, 2024a; Geirhos et al., 2023; Dreyer et al., 2024; Huang et al., 2022). Bereska and Gavves, in their review of mechanistic interpretability for AI safety, highlight polysemanticity as a key challenge in building safer LLMs (Bereska & Gavves, 2024). To bridge this gap, we focus on polysemantic structures in real-world LLMs, particularly those that persist across models, and explore targeted interventions to better understand their associated risks.

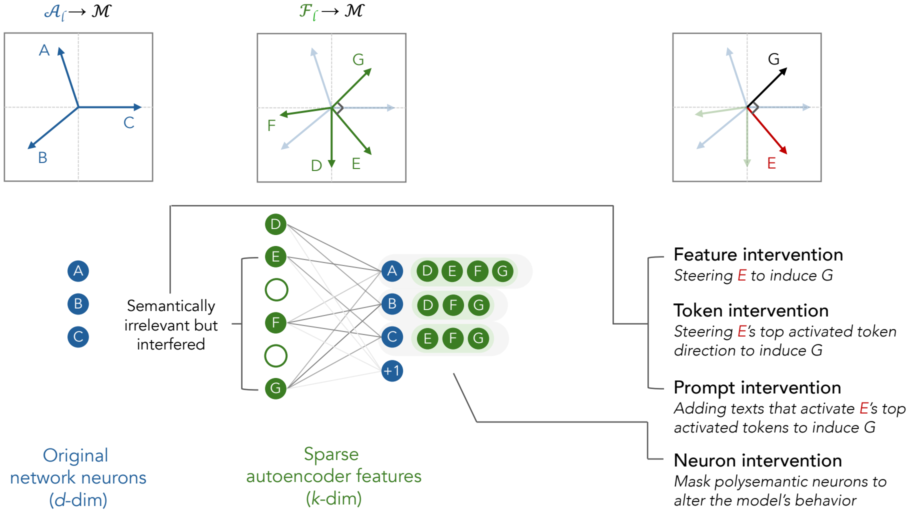

$\mathcal{A}_\ell \rightarrow \mathcal{M}$
A
C
B
$\mathcal{F}_\ell \rightarrow \mathcal{M}$
G
F
D
E
G
E
A
B
C
Semantically irrelevant but interfered
D
E
F
G
A
B
C
+1
D E F G
D F G
E F G
Feature intervention
Steering E to induce G
Token intervention
Steering E's top activated token direction to induce G
Prompt intervention
Adding texts that activate E's top activated tokens to induce G
Neuron intervention
Mask polysemantic neurons to alter the model's behavior
Original network neurons (d-dim)
Sparse autoencoder features (k-dim)

Figure 1: **Conceptual illustration.** Two vulnerable polysemantic structures are described: (1) features are distinct in $\mathcal{M}$ (e.g., E and G) can still interfere in $\mathcal{A}$ and (2) features are often unevenly distributed across neurons (i.e., neuron A encodes more features than B and C).

Before explaining the details, it is necessary to distinguish three nested representational domains:

Human Symbolic Manifold ($\mathcal{M}$): The latent first-order symbolic domain that encodes human-intuitive semantics independent of contextual usage.

Model Activation Space ($\mathcal{A}_\ell$): The d-dimensional vector space spanned by the neurons in layer ℓ of the language model; it partially reflects $\mathcal{M}$.

Sparse Feature Basis ($\mathcal{F}_\ell$): The k-dimensional, typically overcomplete basis ($k \gg d$) extracted from $\mathcal{A}_\ell$ by a SAE.

As illustrated in Figure 1, orthogonality in the activation space $\mathcal{A}_\ell$ does not persist after projection into the symbolic manifold $\mathcal{M}$. Consequently, two features from $\mathcal{F}_\ell$ that appear unrelated in $\mathcal{M}$ (i.e., anchoring semantically to distinct meanings under human interpretation) can still interfere substantially in $\mathcal{A}_\ell$. This interference is also often unevenly distributed across neurons. Building on these two structural characteristics, we design feature, token, prompt, and neuron levels of intervention to investigate: (1) *whether model's expression on a target is sensitive to features and tokens that are semantically unrelated but interfering*, and (2) *whether model vulnerability correlates with neuron polysemanticity, defined as the number of distinct features a neuron encodes*. In this work, model vulnerability to interventions is measured by the shift in the next-token prediction distribution following the intervention.

Our findings are four-fold. First, we present experimental evidence that interventions leveraging polysemantic structures of LLMs can effectively manipulate model outputs. Specifically, by targeting features and tokens—via steering vector techniques—and prompts—via prompt injection—that are not semantically aligned with the intended target but interfere with it, we can reliably induce the model to express the desired semantics. Second, we identify the existence of cross-model persistent polysemantic structures. By collecting shared interference features from both `Pythia-70M` and `GPT-2-Small` and applying them to steer `Llama-3.1-8B/70B-Instruct` and `Gemma-2-9B-Instruct`, we still observe substantial intervention effectiveness, revealing a consistent architecture of meaning that transcends specific implementations. Third, we explore those counterintuitive yet stable interference patterns that replicate across models. Post-hoc annotation of potential higher-order semantic relations accounts for only a minority of cases, indicating that the models learn robust regularities largely opaque to human interpretation. Fourth, we analyze intervention at the neuron level and find that highly polysemantic neurons are more vulnerable: modifying their activation leads to greater semantic shifts in model output. However, for "super-neurons" (i.e., activated by over 500 features) amplification strongly alters model behavior, while deactivation has a notably reduced effect, suggesting they may serve as critical junctions in the semantic architecture.

2 Preliminaries and Methods

2.1 Sparse Feature Extraction with SAEs

Our initial exploration of polysemantic structures draws on the pre-trained SAEs provided by *Neuronpedia*[3]. We focus on `GPT-2-Small` and `Pythia-70M`, the two models for which *Neuronpedia* supplies SAEs for most important sub-modules in every layer. The dimensionality of all the provided SAEs is $32,768$, under which explicit features are extracted. For clarity, in subsequent sections, the direction of a SAE feature is defined as its projection into $\mathcal{A}_\ell$. The interference scale between two SAE features is quantified by the cosine similarity of their projected directions in $\mathcal{A}_\ell$; their surface-level semantic resemblance is measured by the cosine similarity of their projections into $\mathcal{M}$ instead.

2.2 Distinct Feature Identification with Agglomerative Clustering

SAEs disentangle polysemantic neurons into monosemantic sparse features. These features, however, are not always decomposed at a consistent semantic level (Bricken et al., 2023; Foote, 2024). For example, a neuron associated with *dog*-related concepts might be divided into features representing different *dog breeds*, while another neuron encoding both *cat* and *car* concepts might be split into features representing *cat* and *car*. In such cases, the resulting monosemantic features differ in granularity. To mitigate this inconsistency, we employ agglomerative clustering to align feature representations to a consistent semantic level, facilitating both (1) the quantification of neuron polysemanticity and (2) the isolation of feature groups exhibiting low similarity in their surface meanings for subsequent analysis.

To identify distinct, higher-level features, we compute the semantic similarity between pairs of SAE features using their auto-interpretation glosses generated by `GPT-4o-mini` (Caden Juang et al., 2024)[4] together with embeddings of the feature glosses from `text-embedding-3-large`. Prior work offers different heuristics for identifying semantically distinct SAE features. Foote et al. propose a cutoff of 0.5 for distinguishing semantically distinct feature clusters (Foote, 2024), while another work on analyzing `text-embedding-3-large` shows that unrelated concepts typically fall within the 0.05-0.30 cosine similarity range (Zuchen et al., 2025). Drawing on these insights, we conduct agglomerative clustering in each SAE layer using four increasingly strict similarity cutoffs (i.e., 0.40, 0.30, 0.20, and 0.15) to assess whether our results are robust across varying thresholds of semantic distinctness, while retaining sufficient feature density for experimentation. Figure 2 shows an example of the clustering results for the 5th MLP layer of `Pythia-70M` under the similarity cutoff of 0.4. Detailed descriptive statistics on (1) the distribution of cosine similarities among SAE

[3] https://www.neuronpedia.org/

[4] Because different models can yield divergent auto-interpretations of the same feature, we conduct a cross-validation with `DeepSeek-V3` on a single SAE layer of `Pythia-70M`, which reveals both strong concordance between the interpretations produced by `GPT-4o-mini` and `DeepSeek-V3`. Replicating experiments with `DeepSeek-V3` feature glosses further confirmed the consistency of the principal findings. See Appendix I.

features, (2) the distribution of their interference values, and (3) the correlation between interference scale and cosine similarity are provided in Appendix E.

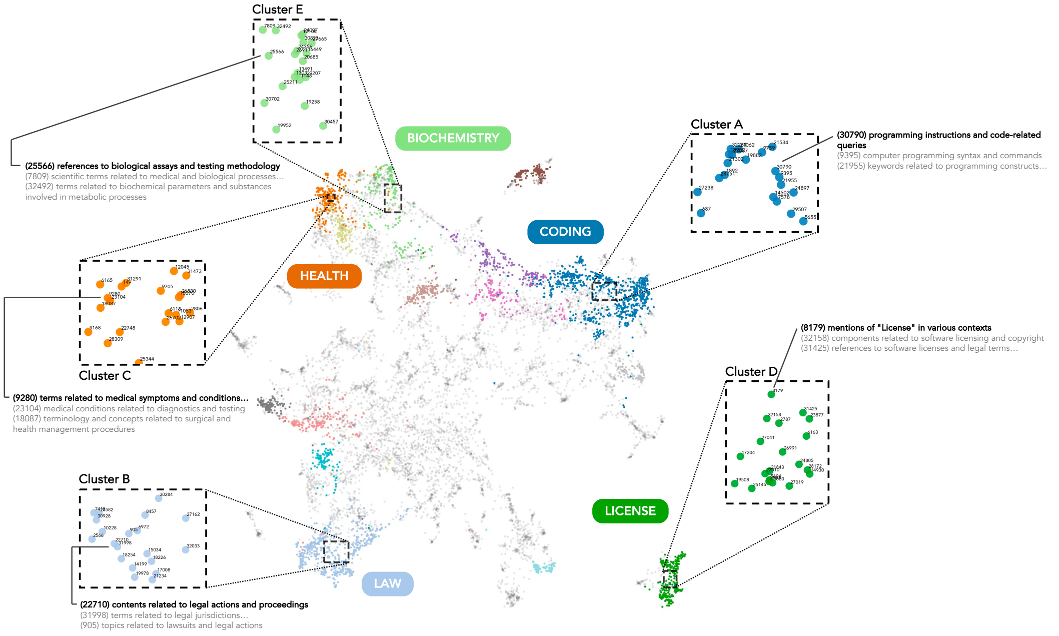


Figure 2: **Agglomerative clustering of SAE features trained on Pythia-70M's 5th MLP layer under the cosine similarity threshold of** 0.4**.** Only the five largest feature clusters are labeled, and the 10 largest clusters are color-coded.

2.3 Dataset Construction

To evaluate the effectiveness of polysemantic interventions in shaping next-token prediction, we construct tailored contextual prompts for each vocabulary token. Specifically, for each token in `Pythia-70M` and `GPT-2-Small`, we use `DeepSeek-V3` to generate three incomplete sentences with varied contexts in which the token is likely to appear next, resulting in a sentence-completion dataset containing $150,000$ prompts for each model. More concrete considerations and details of our dataset generation are stated in Appendix D. In a word, we conduct interventions in specific contexts to influence the output probability of the target token associated with a SAE feature.

2.4 Evaluation Criteria

The success of a polysemantic intervention is quantified as the change in alignment between the model's next-token prediction distribution and a target SAE feature $f \in \mathcal{F}_\ell$. Specifically, we assess the similarity between the model's output and a feature-associated token set $T_f \subset V$, defined as the top-k (here $k = 10$) tokens whose embeddings are most similar to the top-activating tokens for f.

Let $O, \tilde{O} \in \Delta^{|V|}$ be the model's output distributions before and after the intervention, and let $E \in \mathbb{R}^{|V| \times d}$ denote the token embedding matrix. Our main metric is **weighted cosine similarity**:

$$c(P, T_f) = \sum_{t \in V} P(t) \cdot \max_{\bar{t} \in T_f} \cos(E_t, E_{\bar{t}}), \tag{1}$$

where each token $t \in V$ contributes its predicted probability weighted by the highest cosine similarity between its embedding and those in T_f. This captures how semantically aligned the model's output distribution is with the target feature direction. Then, the intervention effect is:

$$\Delta c = \frac{c(\tilde{O}, T_f) - c(O, T_f)}{c(O, T_f)}. \tag{2}$$

As an alternative, we also report **weighted overlap**, which directly sums the output probability mass over T_f. Formal definition and results are provided in Appendix F.

2.5 Overview of Intervention Methods

Our investigation of polysemantic interventions begins with `Pythia-70M` and `GPT-2-Small`, using three complementary approaches: **feature-direction steering**, **token-gradient steering**, and **prompt injection**. We randomly select target features to be intervened[5]. For each selected target feature, we sample interference features from feature clusters derived from Section 2.2—excluding the target's own cluster—to ensure sufficient meaning dissimilarity with the target. Interference features are drawn from five interference intervals: $[0.0, 0.1]$, $[0.1, 0.2]$, $[0.2, 0.3]$, $[0.3, 0.4]$, and $[0.4, 1.0]$. In the first two experiments, we construct steering vectors for interference features using two methods: (1) projecting feature directions from the sparse basis $\mathcal{F}_\ell$ into the model's activation space $\mathcal{A}_\ell$, and (2) computing token-gradient directions from the partial derivatives of the layer's activations with respect to each feature's top-activating tokens. For each intervention, we roughly optimize the scaling of the steering vector over the range $[-20, 20]$, balancing intervention strength with the need to preserve coherent model outputs (i.e., avoiding substantial disruption to the overall output distribution). Details about the tuning strategy are elaborated in Appendix G.2

In the prompt injection setting, we prepend sampled top-activating tokens from interference features to the input and, rather than using weighted cosine similarity, measure how frequently the target feature's top-activating tokens appear among the model's top-10 predictions, conditioned on varying levels of feature interference. To evaluate cross-model transferability, we apply the two scalable intervention methods (i.e., token-gradient steering and prompt-injection) to black-box, larger models `Llama-3.1-8B/70B-Instruct` and `Gemma-2-9B-Instruct`. For these interventions, we use as targets the interference features shared between `Pythia-70M` and `GPT-2-Small`. One might question the practical significance of our next-token control test. While the primary aim of this study is mechanistic, we also include a small-scale evaluation of prompt injection on the *HellaSwag* dataset to assess its ability to steer model behavior in specific target-related cases without compromising general performance. Details are explained in Appendix K.

Finally, we analyze the impact of **neuron polysemanticity** on model outputs in `Pythia-70M` and `GPT-2-Small`. For each neuron, we identify strongly connected features by thresholding connection weights at 0.2, and define its degree of polysemanticity as the number of such features. We then suppress or amplify the activations of neurons with varying polysemanticity levels and evaluate how the model's output distribution shifts toward the semantics of their associated feature clusters.

3 Experiments

3.1 Exploiting SAE Feature Directions for Intervention

Our hypothesis posits that if two feature directions interfere in $\mathcal{A}_\ell$, despite being nearly orthogonal in $\mathcal{M}$, then enhancing one will inevitably influence the other to some degree. If true, this would imply the potential to covertly manipulate the output probability of a target feature by steering with not obviously related features. To evaluate this, we pair each target feature with interference features sampled at varying levels of semantic dissimilarity. We vary feature irrelevance thresholds (from 0.4 to 0.15; Appendix G) to confirm that the observed pattern holds across threshold choices. Additionally, we perform controlled regressions—accounting for feature-pair cosine similarity—under two linear model specifications to confirm that interference effects persist independently of semantic similarity. Figure 3 reports the results, and Table 3 in Appendix G shows particular examples.

All analyses consistently show that steering with features that are semantically dissimilar yet interfering can alter the output probabilities of a target features top-activating tokens, with stronger effects observed at higher interference levels. We also find that SAE-based interventions are generally much less effective on `GPT-2-Small` than on `Pythia-70M`, likely due to the greater depth of the former, which may attenuate the influence of mid-layer activation changes (Fort, 2023).

[5] We randomly sample 480 target features from `GPT-2-Small` and 180 from `Pythia-70M`. See Appendix G for details.

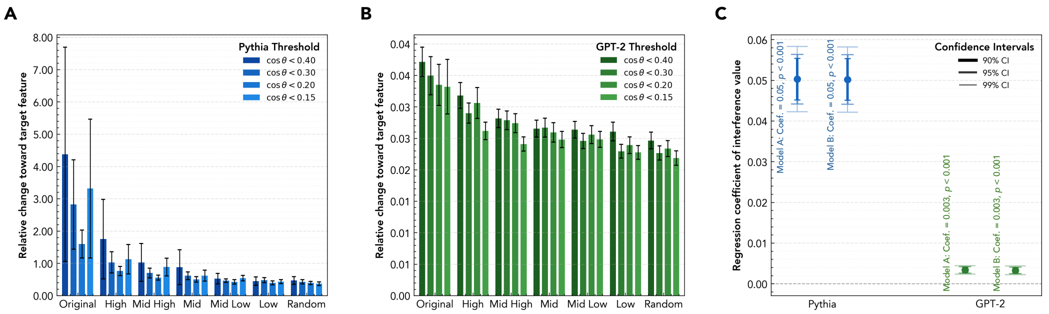


Figure 3: **Effects of intervention based on the interference SAE feature direction.** (A–B) Relative change in weighted cosine similarity toward the target (Δc). Bars show the mean relative change compared to baseline across interference levels, with lighter shades indicating stricter feature-meaning relevancy cutoff thresholds. The x-axis denotes the degree of interference between the target and intervention feature: *Original* corresponds to intervening with the target feature itself, and *Random* serves as a random feature intervention baseline. Error bars denote 95% confidence intervals. (C) Regression estimates of the effect of feature-pair interference value on intervention success. Two regression specifications are shown: Model A regresses weighted cosine similarity after intervention ($c(\tilde{O}, T_f)$) on interference value, with feature-meaning similarity, baseline weighted cosine similarity ($c(O, T_f)$), and layer-type controls; Model B regresses the change score (Δc) on interference value, with feature-meaning similarity and layer-type controls. Error bars denote 90%, 95%, and 99% confidence intervals. Results with the alternative metric are shown in Figure 9.

3.2 Steering with Gradient Vector for Token Intervention

In this section, we treat the top-activating tokens of semantically unrelated yet interfering SAE features as intervention signals (Ferrando et al., 2024). For each interfering feature, we pass its top-activating text through the model and, at the feature's corresponding layer, compute the gradient of that token with respect to all neurons in that layer. This gradient serves as the steering vector. As shown in Figure 4, using token-gradient steering on both models yields roughly $\sim 10\times$ larger effects than steering along SAE feature directions. Interestingly, token-gradient steering also flattens the relationship between interference scale and intervention effectiveness, and steering the original feature's gradient direction is less effective than steering the interfering ones. This may stem from the fact that token gradients are not tied to SAE-defined interference levels and that SAE features can exhibit a degree of arbitrariness (Paulo & Belrose, 2025; Heap et al., 2025).

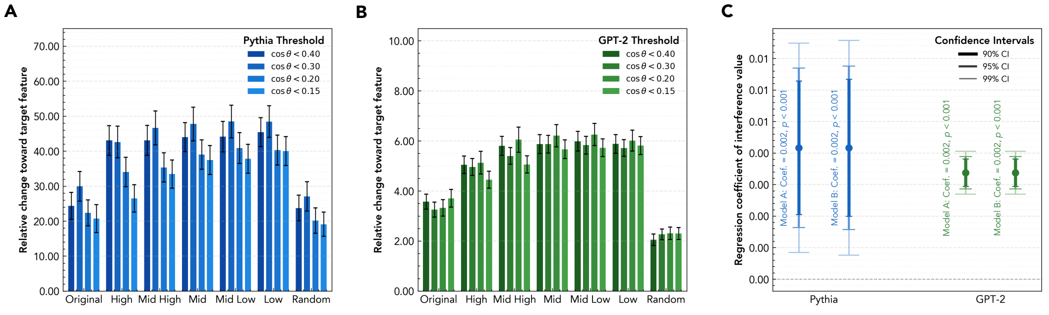


Figure 4: **Effects of interventions along the gradient direction of interference SAE feature's top-activating tokens.** Subpanels follow the same conventions as Figure 3, but intervention vectors are computed from token gradients rather than SAE decoder weights. For (A–B), error bars indicate 95% confidence intervals; for (C), error bars denote 90%, 95%, and 99% confidence intervals. Results with the alternative metric are shown in Figure 10.

3.3 Prompt Injection for Inference Time Intervention

In addition to the two intervention methods described above, which directly modify the model's internal activations, we also investigate whether models are vulnerable to polysemantic interference during inference time. In this case, hidden state manipulation is achieved indirectly by modifying the input prompt. The core idea is that, for a given target feature, we identify highly interfering features, select the tokens with the highest activation values for those features, and inject these tokens into the prompt. This is intended to activate the corresponding features, allowing their influence to propagate and affect the activation of the target feature. The number of ways to inject n tokens into a prompt grows super-exponentially with n, making optimization costly. Here, we apply a straightforward method by prepending 10 randomly selected tokens from each interference value range to the prompt. To assess the generality of our findings, we intervene on the model's production of tokens associated with selected target features, illustrating with three cases: location names, emotion verbs ("hate" or "love") and color-related terms. . For each target set, we identify the corresponding SAE features and extract the top-activating tokens from features with either high or low interference. In addition to these, we include two baseline sets for comparison: a random token set and the original target token set. For each prompt, we sample injection tokens 100 times and compute the success rate as the proportion of runs where target-type tokens are elevated into the top-10 predictions. Table 3 presents representative examples and Table 1 shows macro statistics. As shown, high-interference tokens are more effective at elevating target-related tokens into the top-10 predictions than low-interference or random tokens, though still much less effective than directly prepending the target's tokens.

Table 1: **Comparing intervention effect of prompt injection**

Target	Model	Original	High-interference	Low-interference	Random
Locations	Pythia-70M	65.08%***	36.93%**	32.53%	35.06%
	GPT-2-Small	44.68%***	18.42%***	19.08%***	16.42%
	Llama-3.1-8B-Instruct	33.84%***	20.78%***	19.63%*	18.24%
	Gemma-2-9B-Instruct	10.16%	12.71%***	11.36%	10.66%
	Llama-3.1-70B-Instruct	37.23%***	28.21%***	23.09%**	24.48%
Number	Pythia-70M	65.32%***	35.15%	36.71%***	34.30%
	GPT-2-Small	55.87%***	30.33%	31.23%	34.71%
	Llama-3.1-8B-Instruct	55.97%***	31.87%*	32.90%***	30.57%
	Gemma-2-9B-Instruct	48.42%***	29.93%***	30.64%***	27.16%
	Llama-3.1-70B-Instruct	25.57%***	8.85%***	6.67%	7.09%
Science	Pythia-70M	61.66%***	23.13%	22.78%	28.58%
	GPT-2-Small	75.70%***	25.93%***	26.07%***	21.71%
	Llama-3.1-8B-Instruct	49.67%***	20.08%***	18.95%	17.94%
	Gemma-2-9B-Instruct	46.84%***	20.40%	19.25%	20.15%
	Llama-3.1-70B-Instruct	67.26%***	48.22%***	43.57%	42.24%

Note: Cell values show the success rate of elevating target-type tokens into the top-30 predictions. Gray-shaded rows indicate black-box interventions. Testing uses a shared token set from the two small models. ***, **, and * denote t-test significance at $p < 0.001$, $p < 0.01$, and $p < 0.05$, respectively, vs. random baseline. High- and low-interference tokens lie in $[0.5, 1.0]$ and $[0.2, 0.5]$ for `Pythia-70M`, while $[0.3, 1.0]$ and $[0.2, 0.3]$ in `GPT-2-Small`. Details in Appendix J.3.

3.4 Generalization of Polysemantic Intervention Vulnerability

Since token gradient-based and prompt-based interventions do not rely on pre-trained SAEs, they can be applied to models without access to detailed internal representations. To demonstrate this, we target the `Llama-3.1-8B-Instruct` and `Gemma-2-9B-Instruct` models, reusing the two targets from the prompt injection experiment. We focus on the top-activating tokens of features that interfere with the targets, and identify those shared across both `Pythia-70M` and `GPT-2-Small`. For example, `Pythia-70M` contains 1,309 tokens with interference values above 0.5 for location-related features, while `GPT-2-Small` has 932 such tokens; their intersection includes 193 shared tokens.

We apply token gradient-based intervention to `Llama-3.1-8B-Instruct` and, notably, by extracting steering vectors through random sampling of only one high-interference features' top five activation texts, we could boost the presence of relevant tokens in the top-10 prediction list with over

95% success rate. Prompt injection interventions are tested on both `Llama-3.1-8B-Instruct` and `Gemma-2-9B-Instruct`. As shown in Table 1, high-interference tokens derived from the two small models can steer both larger models more effectively than random baselines. In hindsight, these results suggest that shared polysemantic structures observed in small models also extend to larger models, indicating generalized vulnerabilities that persist across architectures and training regimes.

3.5 Analyzing Shared but Counterintuitive Polysemantic Structure

We observe that many similar feature pairs in `GPT-2-Small` and `Pythia-70M` that are far apart in the first-order symbolic manifold, $\mathcal{M}$ (by surface/gloss semantics), nevertheless lie close in both models' activation space $\mathcal{F}$. To understand this transferability, we first consider mechanisms that couple features via higher-order semantic structure, including semantic-priming-type associations (e.g., thematic/scripts, causal, frame roles) and morphological relatedness, even when overt meanings appear unrelated (Mandera et al., 2017; Bojanowski et al., 2017).

To probe these links, we undertake large-scale annotation using `DeepSeek-V3` and `GPT-5-mini`. The models are prompted to identify higher-order semantic relations for shared interfering feature pairs meeting three filters: interference value > 0.4, semantic similarity in $\mathcal{M} < 0.2$, and cross-model feature-pair cosine similarity > 0.5. In total, 459,229 pairs are annotated (prompt details in Appendix L). Head-to-head comparison of the two model-based annotators is provided in Appendix L. Only 27.7% of feature pairs are judged by at least one model to exhibit potential higher-order associations, and most of those links remain counterintuitive to post-hoc model interpretations. These results both corroborate our filtering strategy for isolating unrelated pairs in the intervention analyses and point to a striking regularity: *LLMs instantiate stable, cross-model polysemantic organization that may be largely opaque to human semantic intuition.* In Appendix L Table 7, we report several examples where models find latent associations between feature pairs and where they do not. One interesting case is the last example marked with an asterisk in Table 7. Both model annotators label it as negative. In our follow-up analysis, however, we hypothesize a deep biographical-affective link: mentions of "Beethoven" may co-occur with expressions of frustration/suffering, given his late-life deafness and celebrated late-period compositions. These examples shed light on the idea that LLM polysemanticity may approximate many latent human knowledge structures; they offer a productive set of hypotheses awaiting rigorous examination. Adjudicating this possibility is beyond our present scope, but we view it as an important direction for future work.

3.6 Manipulating Activations for Neuron Intervention

To complete our discussion, we explore models' vulnerability to interventions on individual neurons. Specifically, we investigate how the degree of polysemanticity in neurons affects the output. For the aggregated features obtained through agglomerative clustering under the cosine similarity threshold of 0.4, we quantify each neuron's connected number of features. Here, we only involve neuron-feature pairs with a connection strength greater than 0.2. Among all neurons, those connected to only one or two aggregated features account for more than 33% in strongly connected neurons, as shown in Figure 5. In addition to neurons connected to multiple or dozens of aggregated features, there are also some "super-neurons" with connections exceeding 500. We examine the impact of manipulating these neurons on the model's output. The experimental results indicate that neurons with higher degrees of polysemanticity are more vulnerable, which means they tend to affect the model's output more effectively. However, for certain "super-neurons," the impact on the model is notably asymmetric: masking them results in even less influence than neurons with lower polysemanticity, while amplifying their activations often leads to exponentially greater effects on model behavior.

4 Discussion

This work makes two main contributions. First, we systematically investigate the vulnerability of LLMs to structured interventions grounded in their polysemantic representations. Specifically, we examine three types of interventions: (1) **feature direction-based**, (2) **token gradient-based**, and (3) **prompt-based**. Feature direction interventions rely on SAEs. While less effective than gradient-based approaches, they form the basis for deriving token gradient vectors. Token gradient-based interventions are more effective and can be constructed directly from activation texts, without requiring SAE pre-training—though they do assume access to internal activations. Prompt-based

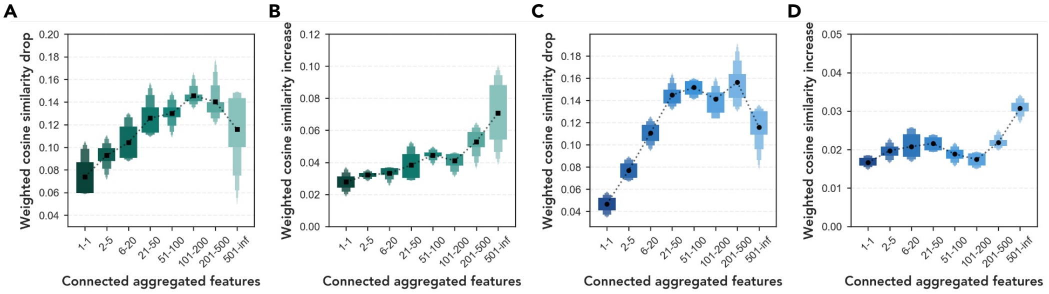


Figure 5: **Effects of activating and deactivating neurons by polysemanticity level.** The x-axis indicates neuron categories grouped by the number of connected features (after clustering). The y-axis reports the change in weighted cosine similarity. Each boxen plot centers on the median (50%) and progressively splits the remaining data in half at each level. A and B correspond to `GPT-2-Small`; C and D correspond to `Pythia-70M`. A and C show the effect of masking neuron activations, while B and D show the effect of amplifying them.

interventions require minimal access and, despite their surface-level nature, still yield meaningful behavioral shifts. Additionally, we explore **neuron-level interventions**, motivated by the uneven distribution of features across neurons. We find that the behavioral impact of masking and amplification correlates with neuron polysemanticity. We also identify a class of "super-neurons," those encoding over 500 features, for which amplification significantly alters model behavior, while deactivation results in a markedly reduced effect.

The second contribution lies in our finding that polysemantic structures identified in two small models transfer to larger, instruction-tuned black-box models (e.g., `Llama-3.1-8B/70B-Instruct`, `Gemma-2-9B-Instruct`) via token- and prompt-level manipulations, producing predictable behavioral shifts without access to internal weights. This suggests that certain polysemantic structures are preserved across architectures and training regimes, exposing a shared representational basis. Notably, this challenges prevailing theories that treat polysemanticity as an incidental artifact of training (Marshall & Kirchner, 2024; Lecomte et al., 2023). Our exploratory analyses further suggest that these transferable polysemantic structures are not reducible to higher-order relations that are readily intelligible to humans; we therefore treat these counterintuitive regularities as testable hypotheses about latent knowledge structure. These results sharpen a central question about LLM polysemanticity: unintended byproducts—or stable, higher-order patterns that await rigorous examination? Finally, our findings strengthen recent evidence of representational consistency and topological stability across models (Huh et al., 2024; Wolfram & Schein, 2025; Lee et al., 2025), even as the origins and functional implications of this consistency remain open challenges.

Our work is the first to systematically evaluate polysemantic structures and vulnerabilities in real-world LLMs, but it has several limitations, including intervention depth and transferability robustness tests. We discuss these limitations and ethical considerations in Appendix N.

5 Conclusion

We systematically probe the sensitivity of LLMs to structured interventions grounded in the polysemantic representations of two small models using SAEs. We show that model behavior can be steered toward specific feature directions by manipulating semantically unrelated yet interfering features via three intervention methods. Interventions distilled from polysemantic structures shared across the small models transfer to larger, black-box instruction-tuned models, indicating a stable and transferable polysemantic topology that persists across architectures and training regimes. Post-hoc annotation suggests that fewer than 30% of these shared interference structures align with higher-order relations readily intelligible to humans; many counterintuitive cases may therefore serve as generators of testable hypotheses about latent knowledge structure. Finally, by leveraging the uneven distribution of features across neurons, we assess models' sensitivity to neuron-level manipulations across degrees of polysemanticity and reveal asymmetric effects in "super-neurons." Together, these findings provide a foundation for future work on the structural properties, vulnerabilities, and representational robustness of LLMs.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits*, 2025.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024. URL `https://arxiv.org/abs/2404.14082`.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

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. *Towards Monosemanticity: Decomposing Language Models With Dictionary Learning*. Anthropic, 2023.

Gonçalo Paulo Caden Juang, Jacob Drori, and Nora Belrose. Open source automated interpretability for sparse autoencoder features. *EleutherAI Blog, July*, 30, 2024.

Ruidi Chang, Chunyuan Deng, and Hanjie Chen. Safr: Neuron redistribution for interpretability. *arXiv preprint arXiv:2501.16374*, 2025.

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

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Maximilian Dreyer, Erblina Purelku, Johanna Vielhaben, Wojciech Samek, and Sebastian Lapuschkin. Pure: Turning polysemantic neurons into pure features by identifying relevant circuits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8212–8217, 2024.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *arXiv preprint arXiv:2406.11944*, 2024.

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. Toy models of superposition, 2022. URL `https://arxiv.org/abs/2209.10652`.

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. Applying sparse autoencoders to unlearn knowledge in language models. *arXiv preprint arXiv:2410.19278*, 2024.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-Jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

Alex Foote. Tackling polysemanticity with neuron embeddings. *arXiv preprint arXiv:2411.08166*, 2024.

Stanislav Fort. Scaling laws for adversarial attacks on language model activations. *arXiv preprint arXiv:2312.02780*, 2023.

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

Robert Geirhos, Roland S Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don't trust your eyes: on the (un) reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Thomas Heap, Tim Lawson, Lucy Farnik, and Laurence Aitchison. Sparse autoencoders can interpret randomly initialized transformers. *arXiv preprint arXiv:2501.17727*, 2025.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Composite backdoor attacks against large language models. *arXiv preprint arXiv:2310.07676*, 2023.

Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. *arXiv preprint arXiv:2202.03423*, 2022.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning*, 2024.

Matthew Khoriaty, Andrii Shportko, Gustavo Mercier, and Zach Wood-Doughty. Don't forget it! conditional sparse autoencoder clamping works for unlearning. *arXiv preprint arXiv:2503.11127*, 2025.

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

Victor Lecomte, Kushal Thaman, Rylan Schaeffer, Naomi Bashkansky, Trevor Chow, and Sanmi Koyejo. What causes polysemanticity? an alternative origin story of mixed selectivity from incidental causes. *arXiv preprint arXiv:2312.03096*, 2023.

Andrew Lee, Melanie Weber, Fernanda Viégas, and Martin Wattenberg. Shared global and local geometry of language model embeddings. *arXiv preprint arXiv:2503.21073*, 2025.

Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*, 92:57–78, 2017.

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

Simon C. Marshall and Jan H. Kirchner. Understanding polysemanticity in neural networks through coding theory, 2024. URL `https://arxiv.org/abs/2401.17975`.

Aashiq Muhamed, Jacopo Bonato, Mona Diab, and Virginia Smith. Saes can improve unlearning: Dynamic sparse autoencoder guardrails for precision unlearning in llms. *arXiv preprint arXiv:2504.08192*, 2025.

Neel Nanda. An extremely opinionated annotated list of my favourite mechanistic interpretability papers v2. In *AI Alignment Forum*, volume 2, 2024.

Tuomas Oikarinen and Tsui-Wei Weng. Linear explanations for individual neurons. *arXiv preprint arXiv:2405.06855*, 2024a.

Tuomas Oikarinen and Tsui-Wei Weng. Linear explanations for individual neurons, 2024b. URL `https://arxiv.org/abs/2405.06855`.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. *arXiv preprint arXiv:2501.16615*, 2025.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. *arXiv preprint arXiv:2503.05613*, 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic, 2024.

Jiachuan Wang, Shimin Di, Lei Chen, and Charles Wang Wai Ng. Learning from emergence: A study on proactively inhibiting the monosemantic neurons of artificial neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 30923103, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671776. URL `https://doi.org/10.1145/3637528.3671776`.

Christopher Wolfram and Aaron Schein. Layers at similar depths generate similar activations across llm architectures. *arXiv preprint arXiv:2504.08775*, 2025.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL `https://arxiv.org/abs/2307.15043`.

Gao Zuchen, Zizheng Zhan, Xianming Li, Erxin Yu, Haotian Zhang, Chenbin Chenbin, Yuqun Zhang, and Jing Li. OASIS: Order-augmented strategy for improved code search. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18451–18467, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.904. URL `https://aclanthology.org/2025.acl-long.904/`.

Content of Appendix

A Related Work

A.1 A Brief Review on LLM Adversarial Interventions

Over the past five years, a growing body of high-impact research has revealed that even aligned LLMs remain vulnerable to a set of converging attack strategies. First, *prompt-space jailbreaks* have evolved from handcrafted exploits into automated, highly transferable methods. For instance, a single gradient-and-greedyoptimized "universal suffix" can consistently bypass refusal policies in ChatGPT, Bard, Claude, and a wide range of open-source modelsdemonstrating both query efficiency and cross-model generalizability (Zou et al., 2023). Second, *activation-space steering* techniques like Contrastive Activation Addition (CAA) show that simple linear interventions in the residual stream can steer behaviors such as hallucination, sycophancy, or toxicity with minimal performance degradation (Panickssery et al., 2023). Third, *parameter-space backdoors*, such as the Composite Backdoor Attack, embed stealthy triggers during fine-tuning that achieve near-perfect malicious compliance without affecting standard benchmarks (Huang et al., 2023). Mechanistic interpretability offers a unifying explanation: transformer activations encode more features than they have dimensions, forcing representations into a compressed superposition and leading to widespread polysemantic overlap (Elhage et al., 2022). Recent work with SAEs has begun to isolate—and in some cases manipulate—these overlapping features directly (Nanda, 2024). Building on this insight, our intervention targets SAE-derived polysemantic directions, integrating prompt-, activation-, and neuron-level interventions into a unified, transferable framework that broadens the known landscape of LLM vulnerabilities.

A.2 A Brief Review on SAE-based Intervention Techniques in LLMs

SAE-based interventions represent a promising direction for developing more interpretable and controllable LLMs. Recent research have introduced a diverse set of SAE-based techniques, such as clamping, patching, and causal tracing, applied across a range of use cases (Farrell et al., 2024; Cunningham et al., 2023; Marks et al., 2024). Empirical results indicate that these methods can be highly effective. For example, targeted unlearning via SAE features has been shown to suppress undesired capabilities with fewer side effects than global fine-tuning (Khoriaty et al., 2025; Muhamed et al., 2025), while feature-level steering enables more nuanced output control than prompt-based methods alone (Rajamanoharan et al., 2024). A key advantage of SAE-based approaches is their efficiency at inference time: they often require only a forward pass with lightweight vector operations and typically do not require model retraining, making them well-suited for real-time interventions.

However, the approach is still in its early stages. Key limitations include challenges in achieving complete and disentangled feature representations, which depend heavily on SAE training quality and selection procedures (Chanin et al., 2024). Computational overhead remains non-trivial, though recent developments such as k-sparse autoencoders and JumpReLU activations offer promising improvements in scalability (Rajamanoharan et al., 2024). There is also a growing need for standardized evaluation benchmarks tailored to intervention methods. A unified benchmark would enable more meaningful comparisons across studies. Currently, researchers often rely on custom evaluation protocols, limiting cross-paper comparability.

In summary, SAE-based interventions offer a powerful mechanism for both understanding and steering model behavior. They uniquely bridge interpretability and utility: not only can we decode model activations into human-interpretable concepts (Cunningham et al., 2023), but we can also use those same features to drive controlled behavioral change (Khoriaty et al., 2025). In this work, rather than focusing on a specific downstream application, we leverage SAEs to investigate structural sensitivities in LLMs—demonstrating that polysemantic features can serve as a substrate for transferable, interpretable interventions. This perspective highlights the broader role of SAEs in the design of more transparent and controllable AI systems.

B Impact Statement

This work systematically investigates a semantic vulnerability in LLMs rooted in polysemanticity—where single neurons encode multiple semantically dissimilar features. We introduce four complementary approaches that expose this vulnerability: manipulating SAE-derived features, token gradients, and prompts to steer model outputs via semantically unrelated inputs, and intervening at the neuron level to reveal a correlation between polysemanticity and output sensitivity. We also identify a class of "super-neurons" whose amplification disproportionately alters model behavior, while masking them has limited effect. These findings not only highlight the unique characteristics of structural fragility of LLMs but also provide practical tools for probing and controlling their internal mechanisms. Our work lays a foundation for future research in AI safety and mechanistic interpretability, not only enabling defenses against such vulnerabilities and more targeted interventions for alignment, but also offering a theoretical lens into the model's internal organization, revealing stable yet counterintuitive interference patterns that may reflect a form of unconscious knowledge association.

C Sparse Autoencoder Training

SAEs are a rapidly developing tool for probing the polysemantic structure of neurons (Shu et al., 2025). Given the activation vector $\mathbf{a} \in \mathbb{R}^{d_{\text{embed}}}$ from a particular model layer, an SAE projects it into a higher–dimensional sparse code $\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}}$ in order to disentangle the multiple semantics that a single neuron may simultaneously encode. The forward computation and the resulting feature definition $\mathbf{f}$ are shown below:

$$\mathbf{f} = \mathrm{ReLU}\big(W_{\text{enc}}\mathbf{a} + \mathbf{b}_{\text{enc}}\big),$$

$$\bar{\mathbf{a}} = W_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}}.$$

The encoder and decoder parameters are

$$W_{\text{enc}} \in \mathbb{R}^{d_{\text{sae}} \times d_{\text{embed}}}, \quad W_{\text{dec}} \in \mathbb{R}^{d_{\text{embed}} \times d_{\text{sae}}}, \quad \mathbf{b}_{\text{enc}} \in \mathbb{R}^{d_{\text{sae}}}, \quad \mathbf{b}_{\text{dec}} \in \mathbb{R}^{d_{\text{embed}}}.$$

The SAE is trained by dictionary learning to minimize

$$\mathcal{L} \;=\; \big\|\mathbf{a} - \bar{\mathbf{a}}\big\|_2^2 \;+\; \lambda \sum_i \mathbf{f_i} \, \big\|\mathbf{W}_{\text{dec}\,[\cdot,i]}\big\|_2,$$

where the first term is the reconstruction loss and the second encourages sparsity (weighted by λ).

For each feature f_i, its direction in the embedding space is defined as the *unit-norm* decoder column. Note that the activation can be represented as linear combination of feature directions. The semantics of features are interpreted by large language models, such as `GPT-4o-mini`, based on their activation texts.

$$\hat{\mathbf{w}}_i \;=\; \frac{W_{\text{dec}\,[\cdot,i]}}{\big\|W_{\text{dec}\,[\cdot,i]}\big\|_2}.$$

D Dataset Generation

When conducting intervention, we do not expect to substantially interfere with every inference of the model, but rather consider intervention in specific contexts, in which we will examine target tokens' prediction probabilities. We require that the constructed sentences be grammatically capable of deriving the target token so that we can check it in top-10 predictions without severely compromising the model. In later sections, we will see that for tokens such as location names and personal names, we can effectively interfere with the output of the corresponding sentences, which shows a potential to produce hallucinations. Also, we need to point out that more specific and general intervention can be achieved by first listing a set of sentences within that context and then identifying possible ways to interfere with each of them. The prompt to generate the dataset is shown below.

```
System: Generate exactly 3 incomplete English sentences where the next word would
clearly be "target_token".  Return a JSON dictionary where:
- The ONLY key is the exact "target_token" (including spaces/capitalization) - The
value is a list of 3 sentence fragments that naturally lead to "target_token"
Example for "target_token=‘ apple’":
{
" apple": [
"She reached into the basket and grabbed",
"The teacher pointed to the red",
"He washed and polished his"
]
}
Rules:
1. All sentences MUST grammatically require "target_token" next to it
2. Use different contexts / scenarios for variety 3. Maintain exact formatting -
no additional keys or explanations
User: target_token={token}
```

We also use DeepSeek-V3 to roughly classify the token types:

```
System: Analyze a token target_token from Pythia/GPT-2-Small's vocabulary and return
its type in EXACT JSON format:
{
"[target_token]": {
"type": "[token_type]",
}
}
Token Type Rules:
1. "verb": Action words (e.g., "run", "jumping")
2. "location": Place names (e.g., "Paris", "Tokyo")
3. "person": Names of people/roles (e.g., "John", "teacher")
4. "object": Physical objects (e.g., "apple", "table")
5. "other_noun": Other nouns not in above categories
6. "adjective": Descriptive words (e.g., "happy", "red")
7. "single_letter": Single characters (e.g., "A", "z")
8. "prefix": Word parts (e.g., "un", "pre")
9. "other": Symbols/punctuation or unclassifiable tokens
User: target_token={token}
```

Here we provide a table of some example tokens and their sentences generated by DeepSeek-V3 (See Table 2).

Table 2: **Token type and prompt sentences examples**

Token	Type	Sentence Examples
London	location	*After a long flight, we finally arrived in* __

Continued on next page

Table 2: **Token type and prompt sentences examples (continued)**

Token	**Type**	**Sentence Examples**
		The train from Paris was heading straight to __ *She always dreamed of visiting the historic city of* __
harbor	location	*The cruise ship slowly approached the bustling* __ *Fishermen gathered at the edge of the protected* __ *The city's economy thrived thanks to its busy* __
Mike	person	*After the meeting, everyone turned to* __ *The teacher called on* __ *She handed the report directly to* __
Trump	person	*The media has been closely following the latest statements from* __ *During the debate, the moderator asked a direct question to* __ *Many supporters gathered outside the venue to catch a glimpse of* __
expert	person	*After years of practice, she became an* __ *The company hired an* __ *When it comes to antique furniture, he's an* __
loves	verb	*She truly believes that everyone* __ *The way he looks at her shows how much he* __ *Despite their differences, their friendship* __
hates	verb	*Everyone knows that she* __ *The way he treats people shows he* __ *It's clear from his expression that he* __
apple	object	*She reached into the bag and pulled out* __ *The smoothie recipe called for one chopped* __ *He carefully balanced the shiny red* __
sad	adjective	*After hearing the bad news, she felt incredibly* __ *The movie's ending left everyone feeling* __ *His eyes told a story of being deeply* __
happy	adjective	*After receiving the good news, she felt extremely* __ *The children were laughing and playing, clearly very* __ *Winning the competition made him incredibly* __

E Supportive Statistics

Active features refer to SAE features that have input texts enabling them to reach an active state. In addition to the semantic clustering of active features mentioned in the main text, we also apply agglomerative clustering to cluster their interference values. The threshold for dividing clusters is set to 0.4. As shown in Figure 7, the vast majority of clusters contain only one feature, indicating that only a small number of features exhibit high interference with others.

In Figure 8, we also provide an overview of the distribution of semantic similarity and interference values between features, as well as their correlations. Since the distribution and correlation of these two types of data remain largely consistent across all layers of the model, we present a representative example layer for illustration.

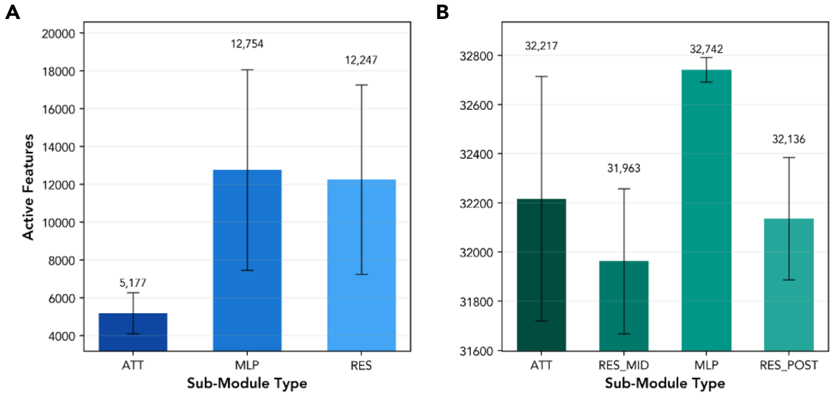


Figure 6: **Number of active features extracted by SAEs per layer.** A is the result of `Pythia-70M`, and B is the result of `GPT-2-Small`. Error bars represent 95% confidence intervals.

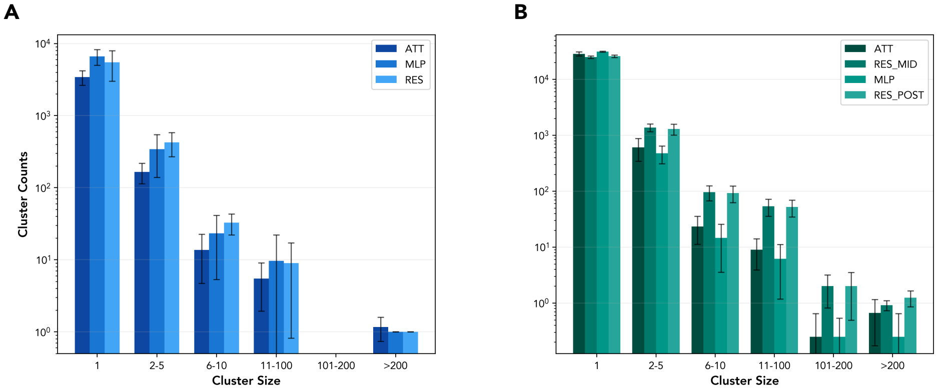


Figure 7: **Interference Cluster Size Distribution.** A is the result of `Pythia-70M`, and B is the result of `GPT-2-Small`. Error bars represent 95% confidence intervals.

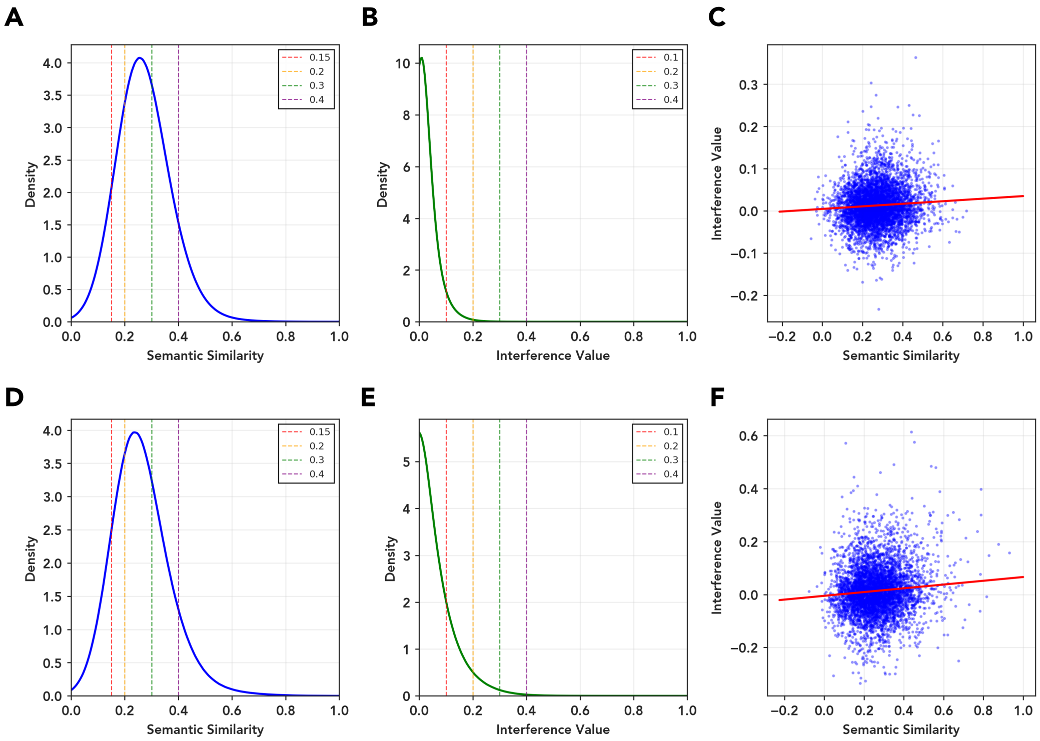


Figure 8: **Semantic similarity and interference distribution of `Pythia-70M Res-4` and `GPT-2-Small Res Post-4` layer.** (A–C) are the results of `Pythia-70M-Res-4`. For semantic similarity, 14.0% of values are below 0.15, 29.6% are below 0.2, 67.4% are below 0.3, and 89.6% are below 0.4. For interference, 85.5% of values are below 0.1, 96.4% are below 0.2, 99.2% are below 0.3, and 99.8% are below 0.4. The bivarite analysis suggests semantic similarity and interference value is positively associated ($\beta = 0.071$, $s.d = 0.092$, $p < 0.001$). (D–F) are the results of `GPT-2-Smal Res Post-4`. For semantic similarity, 10.8% of values are below 0.15, 24.5% are below 0.2, 63.3% are below 0.3, and 89.5% are below 0.4. For interference, 95.6% of values are below 0.15, 99.7% are below 0.2, approximately all values are below 0.3 and 0.4. The bivarite analysis also suggests a positive association ($\beta = 0.030$, $s.d = 0.049$, $p < 0.001$).

F Definition of the Alternative Metric: Weighted Overlap

In addition to the weighted cosine similarity, we report results with an alternative metric, **weighted overlap**, which measures the raw probability mass assigned to the feature-associated token set $T_f \subset V$ in Section G and H. This metric does not smooth over near misses via embedding similarity; instead, it directly captures how much of the model's next-token distribution lands on tokens in T_f.

Definition. For a model output distribution $P \in \Delta^{|V|}$ and target token set T_f,

$$w(P, T_f) \;=\; \sum_{t \in T_f} P(t). \tag{3}$$

Intervention effect. Let O and $\tilde{O}$ denote the model's output distributions before and after intervention, respectively. The (absolute) change in weighted overlap is

$$\Delta w \;=\; w(\tilde{O}, T_f) \;-\; w(O, T_f). \tag{4}$$

Relative change. When a scale-free summary is preferred, we also report the relative change:

$$\widehat{\Delta w} \;=\; \frac{w(\tilde{O}, T_f) - w(O, T_f)}{\max\{w(O, T_f), \epsilon\}}, \tag{5}$$

where $\epsilon > 0$ is a small constant to avoid division by zero.

G Intervention Test with Feature Direction

G.1 Generalized Formulation of a Steering-Vector Intervention

Let $x_{1:T} \in \{1, \cdots, V\}^T$ be the input sequence, $E \in \mathbb{R}^{V \times d}$ be the token-embedding matrix, and $\mathcal{G}_1$ to $\mathcal{G}_L$ be the blocks of a decoder-only Transformer. The unperturbed hidden states are

$$H_0 = E[x_{1:T}], \qquad H_\ell = \mathcal{G}_\ell\left(H_{\ell-1}\right) \qquad (\ell = 1, \dots, L).$$

For any layer index p, we denote the vectorized activation as

$$A_p = \operatorname{vec}\left(H_p\right) \in \mathbb{R}^{d \times T}.$$

With different strategies, we extract the steering direction $z_p \in \mathbb{R}^{d \times T}$. For injection at site s, we define the linear Jacobian:

$$\Phi_{p \to s} : \mathbb{R}^{d \times T} \to \mathbb{R}^{d_s}$$

obtained by composing linear portions between indices p and s. The transported steering direction is

$$z_s = \begin{cases} \Phi_{p \to s} z_p, & \text{if } s > p \\ \Phi^{\dagger}_{s \to p} z_p, & \text{if } s < p \end{cases}$$

where † denotes the Moore–Penrose pseudo-inverse. When $s = p$, we set $z_s = z_p$. Eventually, we modify the activation at site s:

$$\widetilde{A}_s = A_s + \alpha z_s.$$

The network proceeds normally with this perturbation, yielding modified hidden states $\widetilde{H}_\ell$ and logits $\widetilde{y}_{1:T}$.

G.2 Experiment Details

Our hypothesis posits that if two feature directions interfere in $\mathcal{A}_\ell$, despite being nearly orthogonal in $\mathcal{M}$, then enhancing one will inevitably encode some information of the other one into activation space. If true, this would reveal the potential to covertly manipulate the output probability of a target feature by steering the model with seemingly unrelated features.

To obtain the complete intervention data of various levels of interference features on the target features, we first filtered out all features in each layer that contained interference values at all levels, and for which the semantic similarity with the target feature is below the four selected thresholds. Subsequently, we select a subset of these features as the target features and identify the corresponding interference features across the various interference levels. Next, we search for other interference features with low interference values to the target features for experimentation. Specifically, other interference features are selected based on the interference values lying in intervals:$[0.0, 0.1]$, $[0.1, 0.2]$, $[0.2, 0.3]$ and $[0.3, 0.4]$. The scale parameter was tested within the range of $[-20, 20]$, and we avoid larger ranges to prevent severe disruption of the model.

Due to limitations on computational power, we sample clusters and features across various layers. In each SAE experiment with `Pythia-70M`, we sample 180 target features and collect approximately 2,700 interference features. In each SAE experiment with `GPT-2-Small`, we sample 480 target features and collect approximately $7,200$ interference features. For gradient experiments, we reduced 60% sampled features, but three times the number of gradient intervention vectors are extracted to keep the test set scale. As mentioned in the main text, for each feature, we focus on its top-activating token and use `DeepSeek-V3` to generate three prompt sentences for it. For each sentence, we test within the aforementioned scale range and record the result with the greatest improvement in the two metrics. This result means the best performance that the steering vector can achieve to induce the semantics of output toward the target feature without significantly disrupting the model. The final two metrics are averaged across all sentences for all features. To show the robustness of our experiments, the results of the alternative metric are presented in Figure 9.

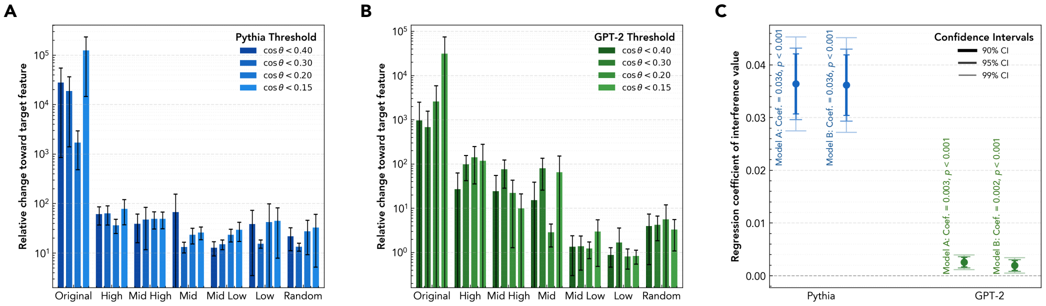


Figure 9: **Feature-level intervention effects measured by weighted overlap.** Subpanels follow the same conventions as Figure 3, but intervention effects are computed using weighted overlap. For (A–B), error bars indicate 95% confidence intervals; for (C), error bars denote 90%, 95%, and 99% confidence intervals.

H Intervention Test with Token's Gradient

H.1 Token Gradient Direction Extraction

Given a tokenized input sequence $x = [x_0, \dots, x_{T-1}]$, let $e_i = E[x_i]$ denote the embedding of token x_i, and $\mathbf{e} = [e_0, \dots, e_{T-1}]$ the full input embedding sequence. Let $f_\ell : \mathbb{R}^{T\times d} \to \mathcal{A}_\ell^T$ denote the models transformation up to layer ℓ. The activation at position i is:

$$a_{\ell,i} = f_\ell(\mathbf{e})[i] \in \mathcal{A}_\ell.$$

We define a scalar probe loss that selects this activation via a linear projection vector $v \in \mathbb{R}^d$:

$$\mathcal{L} = \langle a_{\ell,i}, v \rangle.$$

The gradient of this loss with respect to the input embedding e_i is:

$$g_{\ell,i} := \frac{\partial \mathcal{L}}{\partial e_i} = \frac{\partial \langle a_{\ell,i}, v \rangle}{\partial e_i}.$$

We then normalize this vector to obtain a direction in embedding space:

$$\hat{g}_{\ell,i} := \frac{g_{\ell,i}}{\|g_{\ell,i}\|_2}.$$

We refer to $\hat{g}_{\ell,i}$ as the *token gradient direction*—the direction in input embedding space along which perturbations to token x_i most increase its activation in $\mathcal{A}_\ell$ along v.

H.2 Experiment Details

Steering with the feature direction requires a pretrained sparse auto-encoder of the target model, which incorporates substantial computational costs and lacks scalability. To break this limitation, we need to explore a general approach. Observe that the SAE features are activated mainly by the top-activating token in its activation texts, while other tokens are just diluting its expression. Based on this observation, we can obtain a better steering vector by focusing on this particular token, and a sketch is as follows. We first feed the feature's activation text into the model, then compute the gradients of the top-activating token with respect to all neurons in the layer. The resulting gradients are combined to form a vector.

The SAE dataset from *Neuronpedia* contains approximately 50 activation text segments per active SAE feature, each strongly activating its corresponding feature. Due to computational limitations, we try to extract the gradient vectors from the first 3 activation texts of each feature. Also, we scale the vector within the same range $[-20, 20]$. Additional experimental results are presented below. Each with steering with token gradients combined with steering with feature directions can be done in one hour and a half for `Pythia-70M` and six hours for `GPT-2-Small`, running on a single thread of Intel i7-14700K. The results of the alternative evaluation metric are presented in Figure 10.

Table 3: **Examples of interventions using SAE features, token gradients, and prompt injections**

Type	Model	Intervention	Target feature	Result
Feature	Pythia-70M	Steering feature vector: occurrences of specific surnames	Geographical locations	*“In the next week, we will go to”* ↑ Entered ↓ Dropped Berlin +0.025 our -0.029 London +0.012 some -0.012 To +0.010 an -0.010
	GPT-2-Small	Steering feature vector: positive or negative event outcomes	Expressions of sadness	*“After hearing the bad news, she felt incredibly”* ↑ Entered ↓ Dropped grateful +0.051 bad -0.047 blessed +0.028 guilty -0.044 excited +0.028 uncomfortable -0.025
Token	Pythia-70M	Steering token vector: legal terminology related to licenses and their implications	Elements related to political commentary and critique	*“In the election of this year, it is suggested to vote for”* ↑ Entered ↓ Dropped Donald +0.030 an -0.020 more +0.026 one -0.015 @ +0.015 President -0.013
	GPT-2-Small	Steering token vector: key terms related to prices and transactions	References to location Tokyo	*“The organizing committee just announced that the upcoming finals will be held in”* ↑ Entered ↓ Dropped Tokyo +0.005 Toronto -0.007 Seoul +0.003 Seattle -0.005 Moscow +0.004 London -0.001
	Llama-3.1-8B-Instructed	Steering gradient vector: references to the world and its various aspects	References to ‘Switzerland’	*“I would like to recommend you to spend holidays in”* ↑ Entered ↓ Dropped Switzerland +0.16 Italy -0.034 Germany +0.089 Greece -0.016 Canada +0.015 Bulgaria -0.012
Prompt	Pythia-70M	Injection of the tokens “Court” and “Dat”, both before and within the text	References to locations	*“In the upcoming holiday, we will go to”* ↑ Entered ↓ Dropped Japan +0.021 some -0.014 Europe +0.015 an -0.006 Tokyo +0.012 see +0.003
	GPT-2-Small	Prepending the injection text “(team writers writers)”	Terms related to names or surnames	*“After years of hard work, the award finally went to”* ↑ Entered ↓ Dropped Steve +0.005 China -0.006 John +0.002 waste -0.005 one +0.003 Donald +0.003
	Llama-3.1-8B-Instructed	Prepending the injection text “(placement from placement)”	References to locations	*“In the next weekend we will go to”* ↑ Entered ↓ Dropped Paris +0.011 another -0.013 ** +0.010 H -0.003 - +0.006 K -0.003

Note: ↑ Entered means that corresponding tokens entered the top-10; ↓ Dropped means that corresponding tokens dropped from the top-10. Gray-shaded rows indicate black-box interventions.

I Experiments with `DeepSeek-V3` Feature Descriptions

To evaluate the sensitivity of our findings to the selection of LLMs for auto-interpretation, we employ an alternative model, `DeepSeek-V3`, to interpret the features of a sparse auto-encoder. The explanation text is subsequently processed by the `text-embedding-3-large` model to obtain em-

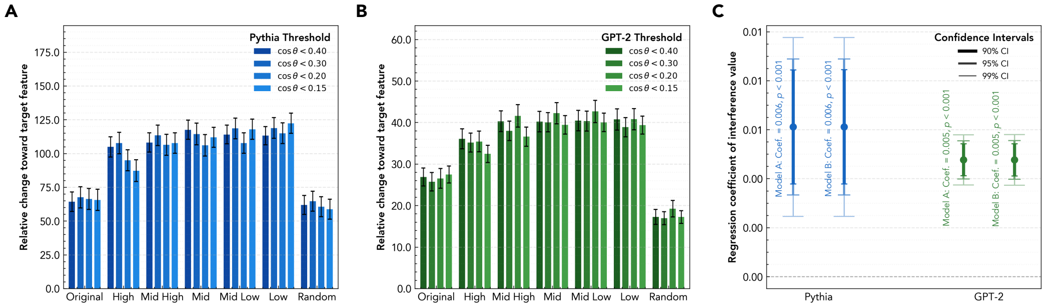


Figure 10: **Token-level intervention effects measured by weighted overlap.** Subpanels follow the same conventions as Figure 4, but intervention effects are computed using weighted overlap. For (A–B), error bars indicate 95% confidence intervals; for (C), error bars denote 90%, 95%, and 99% confidence intervals.

beddings. These embeddings are then utilized as semantic vectors to assess the similarity between features. For this illustrative sampling, features from the `Pythia-70M` att-5 layer are selected. The interpretation of features by large models is generated by feeding the model the top-activating texts of the feature and denoting the high activating tokens in it. Specifically, the prompts we write for `DeepSeek-V3` are as listed below.

System: `We're studying neurons in a neural network. Each neuron activates on some particular word or concept in a short document. The activating words in each document are enclosed by « and ». Look at the parts of the document the neuron activates for and summarize in a single sentence what the neuron is activating on. Try to be general in your explanations. Don't just repeat activation words. Also, you can summarize multiple points if the text content is not highly consistent. Pay attention to things like the capitalization and punctuation of the activating words or concepts, if that seems relevant. Keep the explanation as short and simple as possible, limited to 32 words or less. Omit punctuation and formatting.`
User: `The activating documents are given below:`

```
1.activation_text_1
2.activation_text_2
...
5.activation_text_5
```

In Figure 11, we compare semantic relatedness among SAE features using embeddings of explanations generated by `DeepSeek-V3` and `GPT-4o-mini`. The left density plot shows that pairwise similarities from the two models are tightly aligned, while the right heatmaps further illustrate that both models induce comparable featurefeature semantic structure.

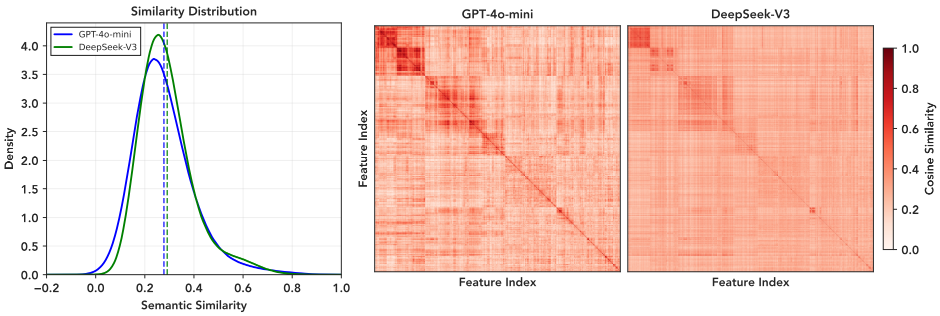


Figure 11: **Semantic relatedness between features from the view of `DeepSeek-V3` and `GPT-4o-mini`.**

Figure 12 compares SAE feature-direction and gradient-based interventions under auto-interpretations from `GPT-4o-mini` and `DeepSeek-V3`. Semantic dissimilarity threshold is set to different scales for intervention feature selection. Although this check is limited to a single layer (and thus exhibits greater variance), we consistently observe that high-interference, low-semantic-similarity features steer the target far more strongly than a random baseline. This pattern holds under both interpreters, indicating robustness to the choice of auto-interpretation model.

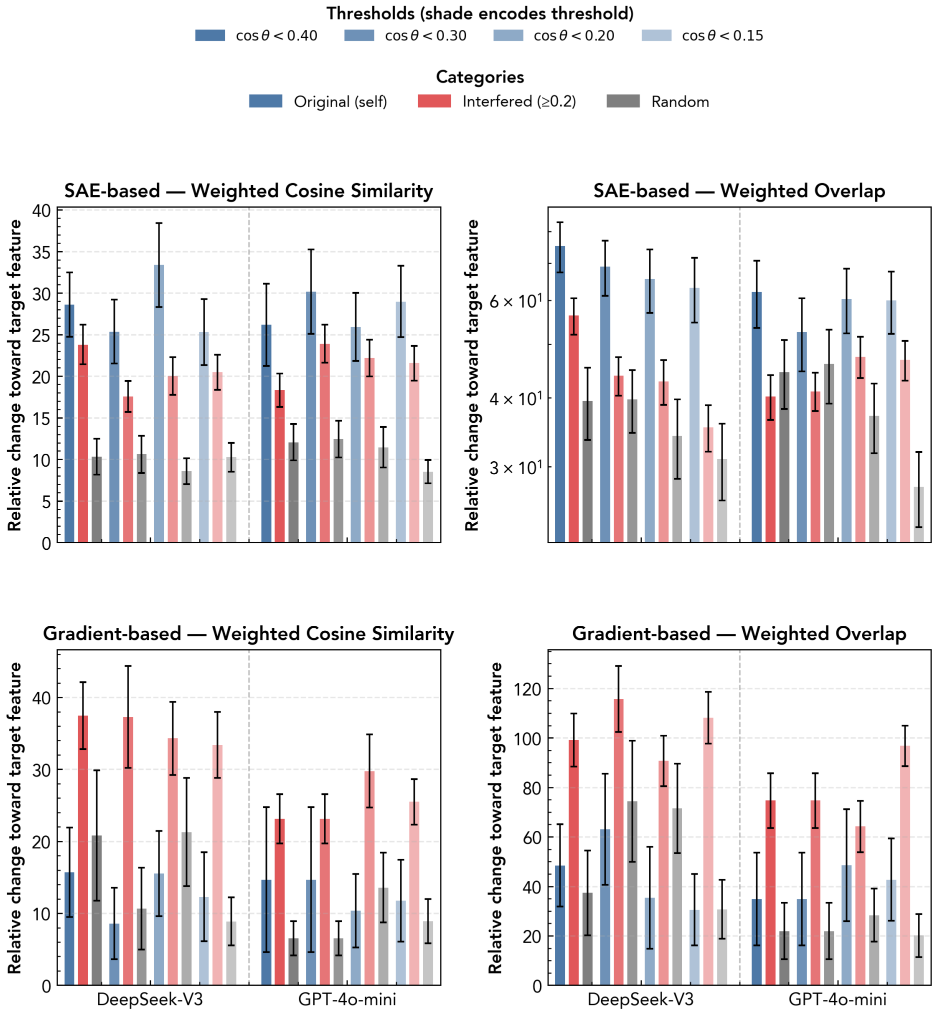


Figure 12: **Intervention effects under `GPT-4o-mini` vs. `DeepSeek-V3` auto-interpretations of SAE features.** Top row shows SAE-direction interventions; bottom row shows gradient-based interventions. Columns are the evaluation metrics: left, Δc; right, $\widehat{\Delta w}$. Error bars denote 95% confidence intervals.

J Black-Box Interventions on Larger Models

We hypothesize that the polysemantic structures learned by large language models may exhibit some degree of generalizability. To explore this further, the interference study is extended to larger models without pretrained sparse auto-encoders.

J.1 Target Selection

We deliberately focus on two feature families—(i) location names and (ii) the polarity antonyms “hate” / “love”—because they satisfy three practical and conceptual criteria that make them ideal first-round probes of polysemantic vulnerabilities.

High corpus frequency, low internal polysemy. Most large-scale text corpora mention both world cities/countries and the verbs hate/love thousands of times, giving the SAE a rich activation signal, yet each term carries a relatively unambiguous core meaning. This minimizes confounds from “target drift” when we measure probability shifts.

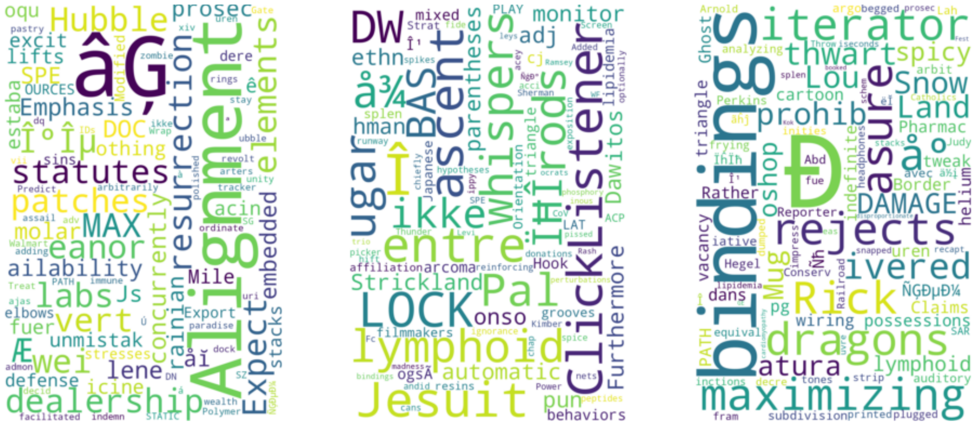
Complementary linguistic classes. Locations are concrete named entities rooted in external knowledge, whereas hate/love are abstract affective predicates that drive sentiment. Showing transferable interference for both a propernoun category and an emotionalvalence category demonstrates that the vulnerability is not restricted to a single part of speech or semantic field.

Policy relevance. Manipulating geographic references risks misinformation about real-world facts (“The capital of X is...”), while manipulating strong sentiment verbs directly impacts toxicity and persuasion. Successful steering of these tokens therefore highlights two distinct, societally significant threat surfaces-factual reliability and affective bias.

Together, these criteria make locations and hate/love a parsimonious yet representative pair for an initial, systematic evaluation; expanding to additional categories is an important next step once the core risk is established.

J.2 Steering with Token Gradient Vector

The scalable intervention on `Llama-3.1-8B-Instruct` is conducted by first selecting target type tokens as mentioned above and identifying target features in `Pythia-70M` and `GPT-2-Small` for which these tokens are the top-activating ones. The two interference feature sets in `Pythia-70M` and `GPT-2-Small` with respect to the target features are then identified. Next, we collect the top-activating tokens in two models respectively, and compute the intersection. The figure13 shows a sketch of the shared interference tokens across two models. From the token cloud map, we can observe that high-interference tokens are often punctuation-like tokens such as line breaks. There are also some tokens with specific meanings, which may be related to certain target tokens in daily context.



(a) Person interference tokens (b) Location interference tokens (c) Emotion interference tokens

Figure 13: **Three types of interference tokens**

After getting the tokens, we proceed to collect the activation texts, which may activate interference features of the target in black-box models. Due to computational constraints, we only compute gradients from and perform operations on the first half of residual layers in `Llama-3.1-8B-Instruct`. The intervention experiments are done for three target token types, each containing about 100 sentences. More intervention examples are listed below. It takes about 20 minutes for a single RTX4090 GPU to find a highly effective gradient vector for steering.

Table 4: **Examples of interventions on Llama-3.1-8B-Instruct Using Token Gradient Vector**

Type	Intervention feature	Result
location	terms related to data and its presentation	*“After months of planning, our road trip finally reached”* ↑ Entered ↓ Dropped New +0.017 an -0.006 Seattle +0.015 it -0.007 San +0.011 our -0.002
	the verb “be” in various forms and contexts	*“She always dreamed of owning a small cafe in”* ↑ Entered ↓ Dropped Vienna +0.080 France -0.008 Munich +0.071 town -0.007 Berlin +0.038 Italy -0.006
	proper nouns, names, and references to specific roles or positions	*“This novel’s opening scene takes place aboard a train to”* ↑ Entered ↓ Dropped Beijing +0.023 New -0.026 Tokyo +0.020 Venice -0.009 Shanghai +0.018 Istanbul -0.010
	quantitative data points related to statistics and performance metrics	*“The rebels established their hidden base deep within”* ↑ Entered ↓ Dropped Afghanistan +0.018 an -0.009 Germany +0.015 their -0.003 Eastern +0.014 one -0.002
	references to specific labeled items or categories	*“His last known coordinates placed him somewhere near”* ↑ Entered ↓ Dropped Paris +0.012 an -0.003 New +0.010 their -0.003 Moscow +0.009 Lake -0.002
name	quantitative data points related to statistics and performance metrics	*“Nobody expected the mysterious package to be from”* ↑ Entered ↓ Dropped Paul +0.261 the -0.104 Emmanuel +0.026 a -0.078 Matthew +0.021 Lake -0.51
	references to academic institutions or concepts	*“The voice on the recording definitely belongs to”* ↑ Entered ↓ Dropped Robert +0.015 a -0.101 Patrick +0.012 the -0.088 David +0.009 me -0.033
	phrases related to pre-approval processes and conditional statements	*“The fingerprints found at the scene match those of”* ↑ Entered ↓ Dropped Michael +0.003 your -0.017 Richard +0.003 one -0.011 Smith +0.004 both -0.008

Continued on next page

Table 4 – **Examples of interventions Using Token Gradient Vector (Continued)**

Type	**Intervention feature**	**Result**
	keywords related to file management and programming constructs	*“This traditional folk song was popularized by”* ↑ Entered ↓ Dropped Bruce +0.010 Pete -0.086 Walter +0.010 American -0.032 Paul +0.007 Woody -0.021
	terms related to multimedia and video production	*“The confidential information was leaked by former employee”* ↑ Entered ↓ Dropped Mike +0.017 and -0.024 Tom +0.012 who -0.024 Bill +0.011 to -0.010
emotion	instances of the verb “is.”	*“After trying the new recipe, my brother absolutely”* ↑ Entered ↓ Dropped love +0.121 fell -0.042 hate +0.095 LO -0.037 dislike +0.015 ad -0.031
	references to legal documents and real estate transactions	*“Science proves that most infants naturally”* ↑ Entered ↓ Dropped Like +0.048 develop -0.099 like +0.020 prefer -0.051 love -0.001 learn -0.034
	phrases indicating topics of discussion or content focus	*“His body language suggests he secretly”* ↑ Entered ↓ Dropped love +0.457 wants -0.158 loved +0.012 enjoys -0.069 hate +0.015 hopes -0.062
	phrases indicating relationships and affiliations in contexts such as surveillance, borders, and regulations	*“This fabric texture makes allergy sufferers”* ↑ Entered ↓ Dropped love +0.126 miserable -0.091 like +0.020 feel -0.074 hate +0.027 and -0.043
	statements that conclude or summarize concepts	*“After the concert critics began to”* ↑ Entered ↓ Dropped hate +0.017 question -0.070 love +0.016 praise -0.064 enjoy +0.010 dissect -0.054

Note: ↑ Entered means that corresponding tokens entered the top-10; ↓ Dropped means that corresponding tokens dropped from the top-10. Gray-shaded rows indicate black-box interventions.

J.3 Prompt Injection

Prompt injection in our experiments is done on the token types listed in Appendix **??**. Based on the classification of tokens, we also generate test sentences for each type, requiring that these sentences grammatically lead to tokens of the target type. The prompt is as follows.

```
System: You are a creative writing assistant specialized in generating incomplete
sentences. Your task is to create incomplete sentences where the next logical token
would very likely be from the category category.

Requirements:
1. Generate approximately 1000 incomplete sentences
2. Each sentence should be approximately 20 tokens long when complete
3. The sentences should end at a natural point where the next word would very
likely be a category word
4. Use diverse contexts, scenarios, and grammatical structures
5. Make the sentences engaging and varied
6. Ensure the incomplete sentences create strong expectation for category words

Examples for category:
- For 'animal': "In the dense jungle, we could hear the roar of a wild"
- For 'color': "The sunset painted the sky a beautiful shade of"
- For 'emotion': "When she heard the news, her face showed pure"
- For 'location': "Our vacation destination this summer will be"
- For 'number': "The recipe calls for exactly"
- For 'person': "The award ceremony will be hosted by a famous"
- For 'science': "The experiment required careful measurement of"
- For 'time': "The meeting is scheduled for next"

Return your response as a Python list of strings, with exactly 1000 sentences.
Format it properly as valid Python code that can be executed.
Start your response with: sentences = [
End your response with: ]

Do not include any explanatory text before or after the list.

User: Generate approximately 1000 incomplete sentences for the category category.

Available type tokens (separated by |, each token is enclosed in quotes):
type_token_1, type_token_2, ...

IMPORTANT NOTES:
- Each token above is a separate vocabulary item from language models
- Some tokens may have leading/trailing spaces (like " dog" or "cat ")
- These are the exact token strings that should be likely to appear as the next
token after your incomplete sentences
- Consider the token boundaries when creating sentences

Please generate diverse, engaging incomplete sentences where the next word would
very likely be from the category category tokens shown above. Make sure to use
various contexts and grammatical structures.

Return as a Python list: sentences = [...]
```

The model returns approximately 400 to 800 sentences for each type. The dataset of token type denotation, along with the example sentences, has been made available in the GitHub repository.

After annotating the tokens in the model's vocabulary with type labels, we partition the token sets based on the intervention information provided by the sparse auto-encoder. First, we filter for features where the highly activated tokens contain the target token type, simply tagging these as target features. Specifically, for each activation text, we identify the token with the highest activation value (max_act) and set a ratio (here, 0.8). Tokens with activation values exceeding ratio * max_act are considered highly activating the interference feature.

We then identify all interference features whose interference value with the target feature exceeds 0.2 and whose semantic similarity is below 0.3. Based on this set of interference features, we further partition them into high-interference and medium-interference sets. Features with interference values above a high_threshold (for a given target feature) are classified as high-interference features, while those with interference values between 0.2 and high_threshold are classified as medium-interference features.

Subsequently, we collect the highly-activating tokens from features in each set to form the respective token sets. All remaining tokens that were not collected constitute the random token set. It should be noted that the high-interference and medium-interference token sets exhibit significant overlap. To address this, we deduplicate the two sets to obtain disjoint token sets. For example, in the case of the `Pythia-70M` model, we annotate 1,938 tokens as belonging to the “location” type. Using the method described above and setting 0.5 as the high-interference threshold, the resulting high-interference token set contains 10,185 tokens, while the medium-interference set consists of 34,946 tokens. And 13,323 tokens remain in the random set. There is an overlap of 9,535 tokens between the high- and medium-interference sets. After deduplication, the high-interference set retains 650 tokens, and the medium-interference set retains 25,404 tokens.

For experiments on `Pythia-70M` and `GPT-2-Small`, we use the token sets generated by each respective model. For experiments on `Llama-3.1-8B/70B-Instuct` and `Gemma-2-9B-Instruct` models, we adopt the union of the token sets from the corresponding interference levels of the two small models. We examine a total of eight categories of token types: location, person, emotion, color, animal, science, number, and time. Apart from the three types mentioned in the main text that demonstrate strong generalizability, the test results for the remaining types are presented below.

Table 5: **Token Types without Strong Generalizability**

Target	Model	Original	High-interference	Low-interference	Random
Person	Pythia-70M	60.28%***	29.02%	28.58%	32.31%
	GPT-2-Small	54.29%***	27.47%***	26.97%**	25.16%
	Llama-3.1-8B-Instruct	38.85%***	19.20%***	19.08%***	16.86%
	Gemma-2-9B-Instruct	43.27%***	25.36%	26.10%	25.04%
	Llama-3.1-70B-Instruct	46.27%***	21.90%***	21.13%**	19.61%
Animal	Pythia-70M	96.96%***	28.49%	30.75%	29.83%
	GPT-2-Small	86.11%***	21.64%	18.87%	25.03%
	Llama-3.1-8B-Instruct	67.07%***	42.76%*	40.56%	40.89%
	Gemma-2-9B-Instruct	26.09%***	41.02%***	38.50%	38.47%
	Llama-3.1-70B-Instruct	49.32%***	32.80%*	30.79%	31.43%
Emotion	Pythia-70M	61.74%***	26.13%***	21.11%	20.68%
	GPT-2-Small	58.84%***	36.91%***	35.22%*	33.85%
	Llama-3.1-8B-Instruct	60.16%***	27.94%	27.45%	29.37%
	Gemma-2-9B-Instruct	56.55%***	13.85%	13.63%	13.50%
	Llama-3.1-70B-Instruct	51.51%***	44.90%***	40.63%	40.40%
Color	Pythia-70M	97.31%***	17.89%	21.49%	21.80%
	GPT-2-Small	85.47%***	35.57%	33.13%	36.93%
	Llama-3.1-8B-Instruct	76.76%***	22.01%	20.67%	20.97%
	Gemma-2-9B-Instruct	22.57%***	13.58%	16.10%	15.67%
	Llama-3.1-70B-Instruct	76.85%***	19.08%	17.88%	18.44%
Time	Pythia-70M	69.92%***	45.33%***	45.54%***	41.33%
	GPT-2-Small	58.48%***	31.55%***	27.72%***	21.42%
	Llama-3.1-8B-Instruct	51.41%***	25.58%	26.29%	25.95%
	Gemma-2-9B-Instruct	55.15%***	25.10%*	25.94%***	23.62%
	Llama-3.1-70B-Instruct	78.28%***	36.87%	36.68%	37.71%

Note: Cell values show the success rate of elevating target-type tokens into the top 30 predictions. Gray-shaded rows indicate black-box interventions. Testing uses a shared token set from the two small models. ***, **, and * denote t-test significance at $p < 0.001$, $p < 0.01$, and $p < 0.05$, respectively, vs. random baseline. High- and low-interference tokens lie in $[0.5, 1.0]$ and $[0.2, 0.5]$. Details in Appendix J.3.

K Covert Intervention on *Hellaswag* Testset

To further examine the impact of the interference vectors obtained in the aforementioned sections on the overall performance of the model, we conduct a rapid validation using the experimental results targeting the "location" type of interference. Given that the intensity of the interference vectors applied in the previous experiments is aimed at maximizing the disruptive effectwhich likely caused substantial impairment to the model, we first reduce the scale to 0.25 times its original value. Subsequently, we randomly select 500 test samples from the *HellaSwag* validation set and evaluate each interference vector on them. Using accuracy as the evaluation metric, the experimental results demonstrate that, out of a total of 174 interference vectors, 149 decreased the accuracy, while 25 increased it. The baseline accuracy is 77.2%, and the average reduction in accuracy is 2.77%. Some examples that keep model performance, i.e. reduce accuracy fewer than 1%, while still having substantial intervention effects are listed below.

Table 6: **Covert Intervention on Hellaswag Dataset**

Intervention Feature	**Top Predicted Tokens**
Description of precautions related to safety protection	*"The documentary crew disappeared while filming in remote areas of"* Raw Intervention the 0.35 the 0.24 Papua 0.027 Papua 0.04 Nepal 0.016 Africa 0.035
Numerical data that may require ordering or sorting	*"She always dreamed of owning a small cafe in"* Raw Intervention the 0.34 the 0.36 a 0.27 a 0.28 her 0.21 Paris 0.13

L Transferable Polysemantic Structure Exploration

In this section, we analyze shared interference patterns in `GPT-2-Small` and `Pythia-70M`, which we previously demonstrated transfer to larger models under intervention. First, we use `DeepSeek-V3` and `GPT-5-mini` to annotate every selected feature pair with the following prompt instruction.

```
System: You are an expert in analyzing neural network feature semantics. Given
two SAE (Sparse Autoencoder) features, their explanations, and their top activation
texts, determine if they are semantically related.
For each feature, you will see:
- An explanation describing what the feature captures
- 5 text segments where the feature activates most strongly. High-activation tokens
are marked with «<token»> or «<multiple tokens»>

Analyze the explanations and the marked tokens/activation patterns to determine:
1. Are these two features semantically related? Consider any form of semantic
relationship - including direct overlaps (capturing similar concepts, linguistic
patterns, or contextual meanings) as well as higher-order associations (e.g.,
semantic priming, thematic relatedness, or complementary roles).
2. If related, provide a concise description (max 50 tokens) of their relationship.
Return your analysis in this exact JSON format:
"isRelated": true/false, "description": "brief description" or null
Examples:
- Features activating on different tenses of verbs: "isRelated": true,
"description": "Both capture verbal expressions, one for past tense, other for
present tense"
- Features for numbers vs. animals: "isRelated": false, "description": null
- Features for positive vs. negative emotions: "isRelated": true, "description":
"Both capture emotional expressions with opposite valence"
- Features for doctor vs. hospital: "isRelated": true, "description":
"Conceptually linked via medical domain (profession vs. location)"
- Features for boat vs. sand: "isRelated": true, "description": "Loosely
associated through beach/marine context (object vs. terrain)"

User: Feature A: feature_a
Explanation: feature_a's explanation
Top activations: texts that activates feature_a the most
Feature B: feature_b Explanation: feature_b's explanation
Top activations: texts that activates feature_b the most
```

To compare the behavior of the two automatic annotators (`DeepSeek-V3` and `GPT-5-mini`), we compute the share of feature pairs labeled as "related" and the simple percent agreement in two subsets. In the `Pythia-70M` subset, `DeepSeek-V3` labels 10.8% of pairs as related and `GPT-5-mini` 28.7%, with 80.3% percent agreement. In the `GPT-2-Small` subset, the corresponding proportions are 3.7% (`DeepSeek-V3`) and 18.4% (`GPT-5-mini`), with 83.8% agreement. Aggregating both subsets, `DeepSeek-V3` labels 9.5% of pairs as related and `GPT-5-mini` 26.8%, and the overall percent agreement was 80.9%. We define percent agreement as the proportion of pairs for which both annotators assign the same label. Figure 15 A-B shows the agreement matrices for the two annotators. Interestingly, the overall consistency between the two annotators is low (`Pythia-70M`: Cohen's $k = 0.408$; `GPT-2-Small`: Cohen's $k = 0.221$). Compared with `DeepSeek-V3`, `GPT-5-mini` labels significantly more feature pairs as related, suggesting greater sensitivity to latent semantic associations. Nevertheless, more than 70% of pairs are still judged completely unrelated in the annotators' combined judgment (i.e., both say unrelated).

In Table 7, we further report eight qualitative examples where the two annotators flag latent associations and contrasting cases where they do not.

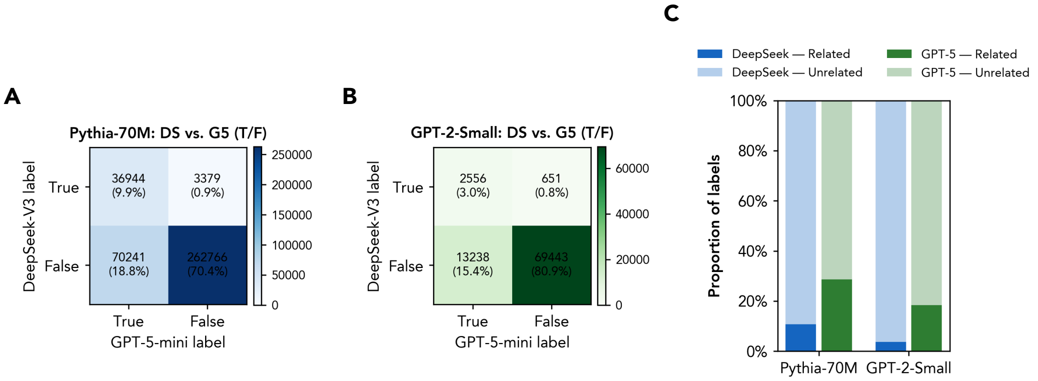


Figure 14: **`GPT-5-mini` and `DeepSeek-V3`'s annotation reports for shared interfered feature pairs.** DS is the abbreviation of `DeepSeek-V3` and G5 is the abbreviation of `GPT-5-mini`.

Table 7: **Eight examples of annotated feature pairs.**

Category	Feature pair	Interpretation
Frame	**Feature A:** Phrases and sentences that highlight systemic issues related to incarceration and its effects on individuals and families. **Feature B:** Quantitative information related to statistics and predictions.	Incarceration content often appears alongside quantitative/statistical references. (from `GPT-5-mini`)
Orthography	**Feature A:** Occurrences of the word "acher." **Feature B:** References to specific researchers or authors in studies related to minimum wage.	Both capture researcher/author name substrings (e.g., -acher / Wascher). (from `GPT-5-mini`)
Axiology	**Feature A:** The term "dear" in emotional contexts related to relationships and feelings of affection. **Feature B:** Concepts related to the notion of the sacred.	Both capture emotionally significant concepts (affection vs. sacredness) with deep personal or spiritual value. (from `DeepSeek-V3`)
Homography	**Feature A:** References to playing cards and card-related concepts. **Feature B:** References to the authors or studies related to economic analysis.	Surface lexical overlap: "card" as playing card vs "Card" (author name). Same token, different senses. (from `GPT-5-mini`)
No-relation	**Feature A:** References to "New Guinea." **Feature B:** File formats and file compression terminology.	None
No-relation	**Feature A:** Instances of the abbreviation "ob" or related terms indicating observational data or annotations. **Feature B:** A specific term related to a well-known ride-sharing company.	None
No-relation	**Feature A:** Terms related to turbidity and its measurement. **Feature B:** References to specific music artists or groups.	None
No-relation*	**Feature A:** Occurrences of the name "Beethoven" and related variations. **Feature B:** Words related to expressions of frustration or annoyance.	None

M Polysemantic Neuron Manipulation

During the examination of SAE features and their connections with neurons, many features exhibit semantically similar activation texts. To avoid repetitive analysis on similar activation texts, we first perform feature clustering based on semantics of activation texts, and then check neuron connection at the cluster level. Given that the sparse auto-encoder from *Neuronpedia* is trained with a sparsity setting of 3, the analysis focuses on the top three neurons with the highest alignment values per cluster. A threshold of 0.2 is applied to filter out weak connections. Figure15 shows the distribution of polysemantic neurons identified in each layer. We can see that polysemantic neurons with strong connections with aggregated features only take up fewer than 5% in each layer.

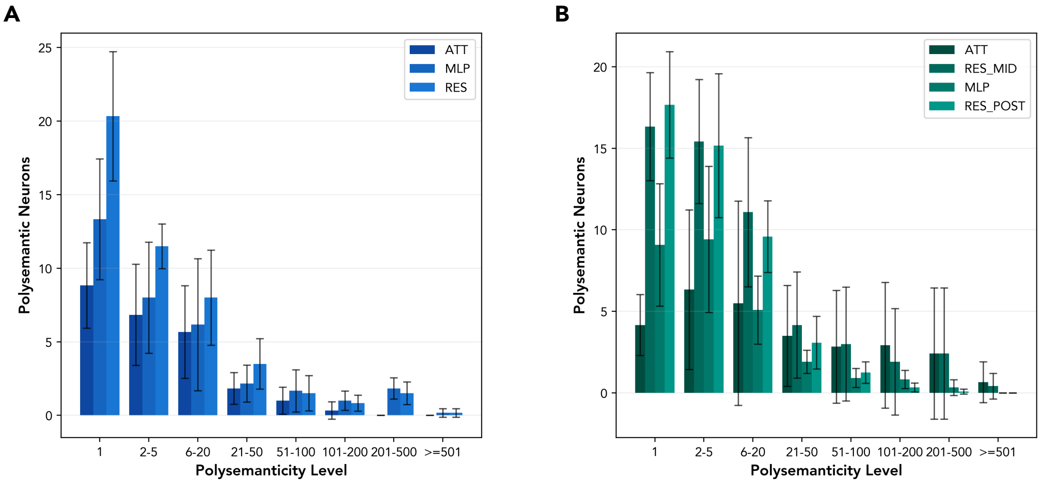


Figure 15: **Distribution of polysemantic neurons in each model.** A is the result of `Pythia-70M`, and B is the result of `GPT-2-Small`. Error bars represent 95% confidence intervals.

For strongly connected polysemantic neurons, we do further investigations on how suppressing or boosting their activation influences the semantic shift in the model's output to their aligned features. Neurons' activation is multiplied with a scale value in the range $[0, 20]$. Note that scaling within $[0, 1]$ suppresses activation, while scaling within $[1, 20]$ amplifies it.

N Ethics Statement, Limitations and Future Works

This study has three key methodological limitations. First, we rely on SAEs to disentangle polysemantic activations; although SAEs are the de-facto tool, their outputs fluctuate with dimensionality and hyperparameters, yielding unstable features (Paulo & Belrose, 2025; Heap et al., 2025; Gao et al., 2024). Second, our interventions steer only one interference feature in one layer, while multi-feature, cross-layer manipulations could amplify and better obscure the effect (Ameisen et al., 2025). Third, we quantify vulnerability solely via shifts in immediate next-token probabilities on two small base models—because only they both expose raw logits and have pre-trained SAEs—then check coarse transfer on three larger instructed models; establishing how these interventions alter non-trivial downstream tasks in bigger models is the next stage of this project. We release complete code, evaluation scripts, and synthetic data here.